Artificial Neural Networks and its Integration with Genetic Algorithm and Fuzzy Logic

A Thesis Submitted
in partial fulfillment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By

ADHIKARI AMITABH PRASAD

To the

NUCLEAR ENGINEERING AND TECHNOLOGY PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
JULY 1995

NETP-1995-M-PRA-ART

CERTIFICATE

It is to certify that the work contained in this thesis entitled "**Artificial Neura Networks and It's Integration with Genetic Algorithms and Fuzzy Logic**" b *Adhikari Amitabh Prasad*, has been carried out under our supervision and th this work has not been submitted elsewhere for a degree.

Prem K Kalra
(Dr P.K. Kalra)
Associate Professor
Dept. of Elec. Engg.
I. I. T. Kanpur

K. Sri Ram
(Prof. K. Sriram)
Professor
Dept. of Nuclear Eng
IIT Kanpur

ACKNOWLEDGMENT

I feel extremely happy to express my sincere gratitude towards Dr. P. K. Kalra and Prof. K. Sriram, my thesis supervisors, for introducing me to the wonderful world of Artificial Neural Networks and different AI techniques such as fuzzy logic, genetic algorithms. The open atmosphere for exchanging the views with him, coupled with his constant encouragement right from the beginning, was the main driving force throughout this work. My special thanks goes to Dr. Rajeev Shekher, Asst. Prof., MME Dept., for providing me the concepts and ideas in the second part of this thesis, application to *Pachuca Tank*, without which this work could never have come to an end. A number of discussions with him has been of considerable help to me in this work.

I am grateful to my friends, mainly to Mr. Basant K. Kukerja , for helping me in development of ANNS *the simulator* and providing his technical help from time to time. I also want to thank Mr. G. G. Roy for providing his experimental data and sharing his experience for pachuca tank problem. Last, but not the least, I would like to thank Vidhu bhabhi for her constant encouragement at odd times.

Adhikari Amitabh Prasad

ABSTRACT

Building Intelligent System that can model human behavior has captured the attention of world for years. Different Artificial Intelligent techniques such as Artificial Neural Networks, Genetic Algorithms, Expert System, Fuzzy Logic, Learning Automata are originated due to such interest. These Algorithms represents different level of human information and hence not complete in itself to represent human brain. These clues indicate that integration of these paradigms may perform better for real world problems compared to an independent paradigm. In this work, an attempt has been made to integrate different Artificial Intelligence Paradigms with Artificial Neural Networks. The attempt has also been done on improving different Backpropagation (BP), one of the most popular artificial neural network algorithm.

In the first stage, the BP and it's modification was developed. Radial Basis Function (RBF) is developed in the second stage to reduce the learning time. In the third stage integration of ANN is done with genetic algorithms to obtain generalized weights of ANN. Integration of ANN with fuzzy logic is done in fourth stage. Lastly, ANN is integrated with genetic algorithms and fuzzy logic together.

The performance of different types of these variation is tried on two real world problems one from power system (Short Term Load Forecasting) and second from nuclear engineering (Oxygen Mass Transfer in Air Agitated Pachuca Tank). The results indicate that integration of ANN with genetic algorithms and fuzzy logic gives the best result in short term load forcasting. Again success of BP's variation is reflected in the case of Pachuca Tank problem. RBF indicates better choice than BP where fast learning is required. There is further scope of improvement of result by fine tuning some of the parameters.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Several attempts have been reported to understand and model the capabilities of human brain. Some of these are *Expert System* [1], *Genetic Algorithm (GA 's)* [2, 3], *Artifical Neural Network(ANN)* [4], *Fuzzy logic* [6], *Learning Automata* [5], *Induction tree* [7] and *Cellular Automata* [8]. These algorithms represents different level of human information processing. It has been believed that human mind has exceptional capability to *recall, optimize, memorize, sort* and *search.* However, all functions performed by brain may not be performed by each model independently. Hence, the **integration** of these paradigms of Intelligence may perform better for real world problems compared to an independent paradigm. A brief discussion of some of the above paradigms is presented in the following paragraphs.

1.1 Artificial Intelligence Techniques

1.1.1 Expert System

Expert system based system encodes the knowledge of human expert. An Expert system consists of a *Knowledge base* which is designed with the help of a *human expert* and consists of expert level information necessary to solve problems. This information is rep-

resented in several ways [1], but more commonly as *a set of rules.* These rules usually takes the form *If* < *if part* > *Then* < *then part* >. The *if-part* of the rule is commonly referred to as its left hand side, premise or antecedent. Premises contain a collection of conditions that must be specified before the rule may be used. The *then part* of the rule (its right hand side, conclusion, or consequence) contains a set of actions to be performed when the rule is applied. Thus the performance of Expert System depends upon *knowledge acquisition representation* and *decision models.*

Expert System are inherently *domain specific*, hence non transferable. The **knowledge acquisition** is difficult as in most of the cases due to *patient rights, confidentiality* and *fear of replecement by computers.* Further, conflict resolution, addition of rules, and modifications in decision models are cumbursome and *time consuming* making the scope of expert system very limited. The process which is goverened by *declarative knowledge* can easly be modelled by this paradigms.

1.1.2 Artificial Neural Networks

ANN consists of many simple, elements called *neurons.* The neurons interact with each other using weighted connection similar to **biological neurons.** Inputs to artificial Neural Net as multiplied by corresponding weights, All the weighted inputs are then segregated and then subjected to non-linear filtering to determine the state or active level of the neurons.

Neurons are generally configured in regular and highly interconnected topology in ANN. For example, in the Hopfield model [9], neurons form a fully connected topology, and output from each neuron feeds as an input to the neighboring neurons. In the backpropagation model [10] and Boltzmann machine [9], the networks consist of one or more layers between input and output layers. In the self-organizing feature map [11], the networks connect a vector of input neurons to a two-dimensional grid of output neurons. There is no clear cut methodology to decide parameters, and topologies of ANN and type of training, hence to build the ANN is time consuming and computer intensive. However these can be used in real time because of *inherent parallelisms*, and *noise immunity characteristics.*

1.1.3 Fuzzy Logic

Fuzzy logic is a *super set* of conventional (Boolean) logic that has been extended to handle the concept of partial truth – truth values between *completely true* and *completely false.*

According to Zadeh [6] one should not visualize fuzzy theory as a single theory, but one should regard the process of **fuzzification** as *a methodology to generalize ANY specific theory from a crisp (discrete) to a continuous (fuzzy) form.* This technique is very helpful in modeling problems which consists of *linguistic variables* like very small, small. Fuzzy theory works on *membership functions* [12] and their *operation* to obtain a conclusion. The membership function of a variable represents its *degree of truth.* In order to obtain different conclusions different fuzzy operators (*analogous to boolean logic's And and Or operators*) are used . Although fuzzy logic showed it's strength in many fields [13]. It suffers from some limitations. The transformation of human knowledge into fuzzy knowledge base for approximate reasoning is time consuming because modification is done in fuzzy knowledge base untill the accepatable levels of results are obtained. Even the method for tuning the membership function does not exist. Furthermore the fuzzy decision modeling become very complex when the number of variables is very large (causes too many number of rules). This paradigm is very effective in handling useful linguistic uncertainty and which has multiple solutions.

1.1.4 Learning Automata

Learning Automata is also one of the AI technique. This can be qualitatively represented as follows : A finite number of actions can be performed in a random environment. When a specific action is performed the environment provides a random response. which is either favorable or unfavorable. The system is corrected until it achieves a state which has high *probability* of producing a favorable response. It may not be possible to for same problems to achieve desired response with probablity close to zero. More over for a very complex problem this technique is very expensive as it requires a **large computer storage and is computationally intensive** [5]. However this technique offers advantages of probablistic

system and higher order system that is, collection of neurons can be thought of as single learning automata.

1.1.5 Genetic Algorithms

Genetic Algorithms are based on models of *natural adaptation* and *evaluation.* These learning systems improve their performance through process which model population genetics and survival of the fittest. The processes of GA's depends mainly on three operators : crossover, mutation and selection , based on natural evaluation. This optimization technique has low probability of getting trapped in local minima than traditional optimization technique [16]. GA's do not use gradients and hence are suitable for those systems where evaluation of the derivative is difficult. It has in built parallelisms and can search all global optima. This process may give better results than traditional methods, unfortunately it is very **time consuming**. Again selection of parameters in GA's play very important role for it sucess.

1.1.6 Induction Tree

Induction tree is one of the possible approach to multistage decision making. The basic idea involved in this approach is to split a complex decision into simpler decisions, with the hope that final solution obtained in this method would resemble the desired solution. For simple, less complex problems this method works fairly well. Large number of classes, however, **increase the search time and the probability of overlap.**

1.2 Integration of A.I. Techniques

The brief description of Intelligent system paradigms indicates that each paradigm has built in tolerences. Hence, to overcome the limitations one needs to integrate these paradigms. It is possible to integrate two or more paradigms together to achieve the system which exhibits relatively better performance in the terms of learning, response time, computational requirements, and complexity of implementation.

Fuzzy system and neural networks are both capable of handling **uncertainty and imprecision** and have found many successful applications [14]. Neural networks can be mapped with the help of fuzzy logic [15]. In some cases even weights of the ANN can be made fuzzy and different decisions can be made with the help of different fuzzy operators. In some cases fuzzy preprocessing or post processing of ANN data set can be done.

Genetic Algorithms are also used to integrate fuzzy as well as artificial neural systems. In some cases membership function can be decided with the help of genetic algorithms. GA's can be helpful in deciding topology as well as weights of ANN [27]. Decision tree are also used to determine the optimal topology of ANN. Fuzzy system can also be integrated with GA's where different probability based operation of GA's can be replaced by different fuzzy operators based operation.

1.3 Objective

Integration of different AI techniques has shown improvement over independent paradigms [17]. In this thesis different AI techniques such as **Fuzzy Logic , Genetic Algorithms** are integrated with **ANN**. The resulting algorithms are then tested against data from the problems connected with power and nuclear ndustries.

1.4 Organization of Thesis

Bottlenecks of Feedforward ANN it's solution are discussed in Chapter 2. The simulator **ANNS – Artificial Neural Net Simulator** which is develop to overcome some of the problem is discussed in Chapter 3. Chapter 4 contains the results of two different test problems obtained from the simulator. Finally conclusion and future scope of the works are discussed in chapter 5.

Chapter 2

Bottlenecks of Feedforward Models ANN And Its Possible Solution

Most method of training multi-layered artificial neural networks (ANN) are based on the steepest descent technique [4], which frequently have problems such as very poor convergence behaviour, trapping in local minima, misdirection of decent, and oscillaion. Furthermore the ANN also suffers from problems like network paralysis, overgeneralization, and multiple solution problems. Therefore the objective of the chapter is to discuss the methodologies for overcoming the above problems.

Development of ANN is performed in two phases:

1. **Training phase**: Here process ANN tries to memorise the pattern of the learning data set. This phase consists of the following modules:

   - **Selection of neuron characteristics**
   - **Selection of Topology**
   - **Error minimization procedures**

- **Selection of training pattern and preprocessing**
- **Stopping Criteria of training**

The above mentioned modules are elaborated in the next sections.

2. **Testing phase**: Here ANN tries to predict and/or test data sets.

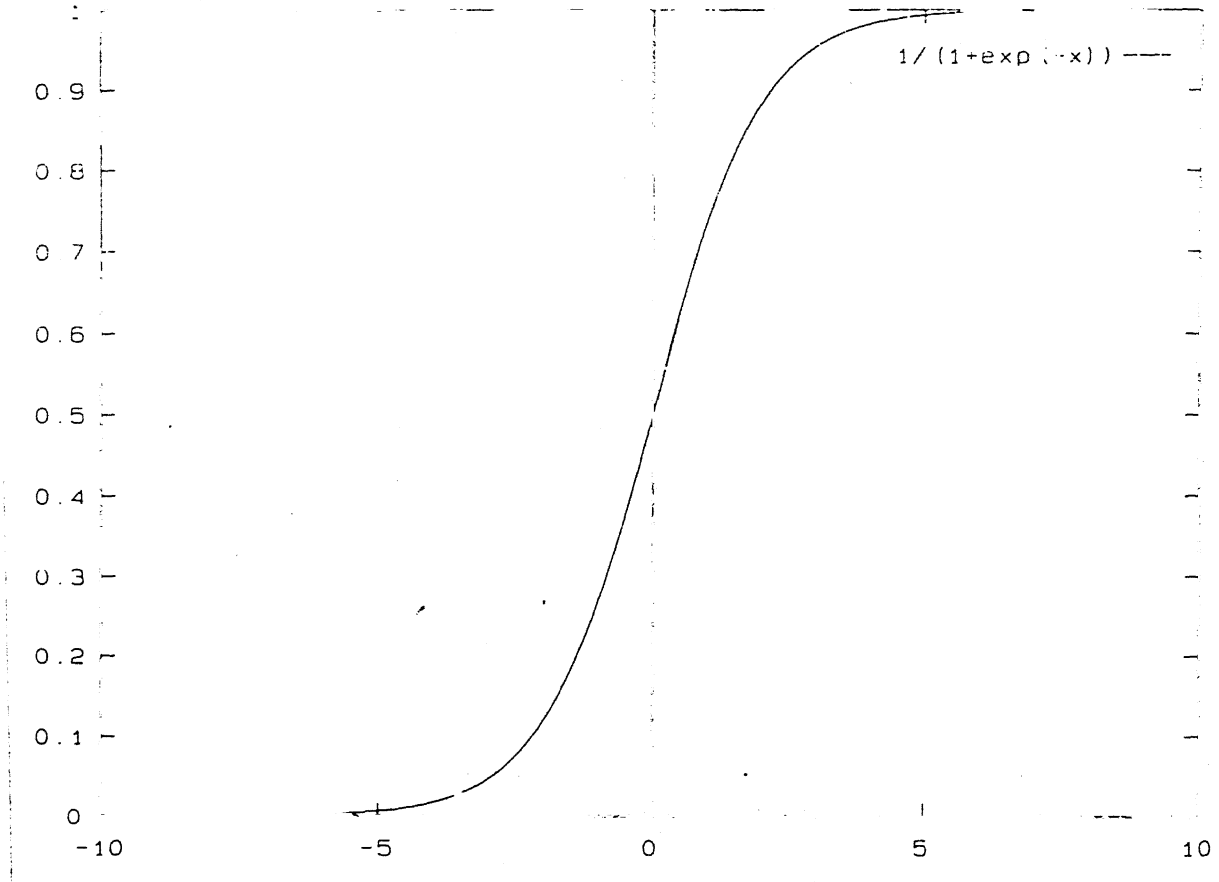
2.1 Training phase issues

2.1.1 Selection of neuron characteristics

Neurons can be characterized by two operations aggregation and nonlinear filtering. Non linear filtering (some time called activation function) can be characterized by several function , sigmodial and tangent hyperbolic functions [figure 2.1] being the most common. Both have similar transfer characteristics and hence mapping properties [18]. Even though numerous activation functions have been reported in literature [19], care should be exercised in selecting the activation function which are problem dependent. For example, the Logarithmoid activation as shown in Figure 2.2 function is used when the upper limit is unbounded while the *Radial Basis Function (RBF)*[18] is used for problems having complex boundaries. The main objective is to design an ANN of superior generalization ability and fewer number of nodes. This avoids unnecessary calculations and it can be used for fast decision making [20] - [23].

Certain class of problems can be suitably handled by fuzzy networks [24]. In fuzzy networks some nodes are made up of membership function (similar to activation function in Backpropagation), π and S type [15]. The selection of membership function is again problem dependent and can be handled accordingly.

Since most of the ANN use the gradient method to minimize error, points where the gradients are small should be avoided in the beginning of the process (small process makes the learning process slow). Therefore initialize the learning process, the central part of the function where the gradients are large is selected. This can be achieved by scaling proper



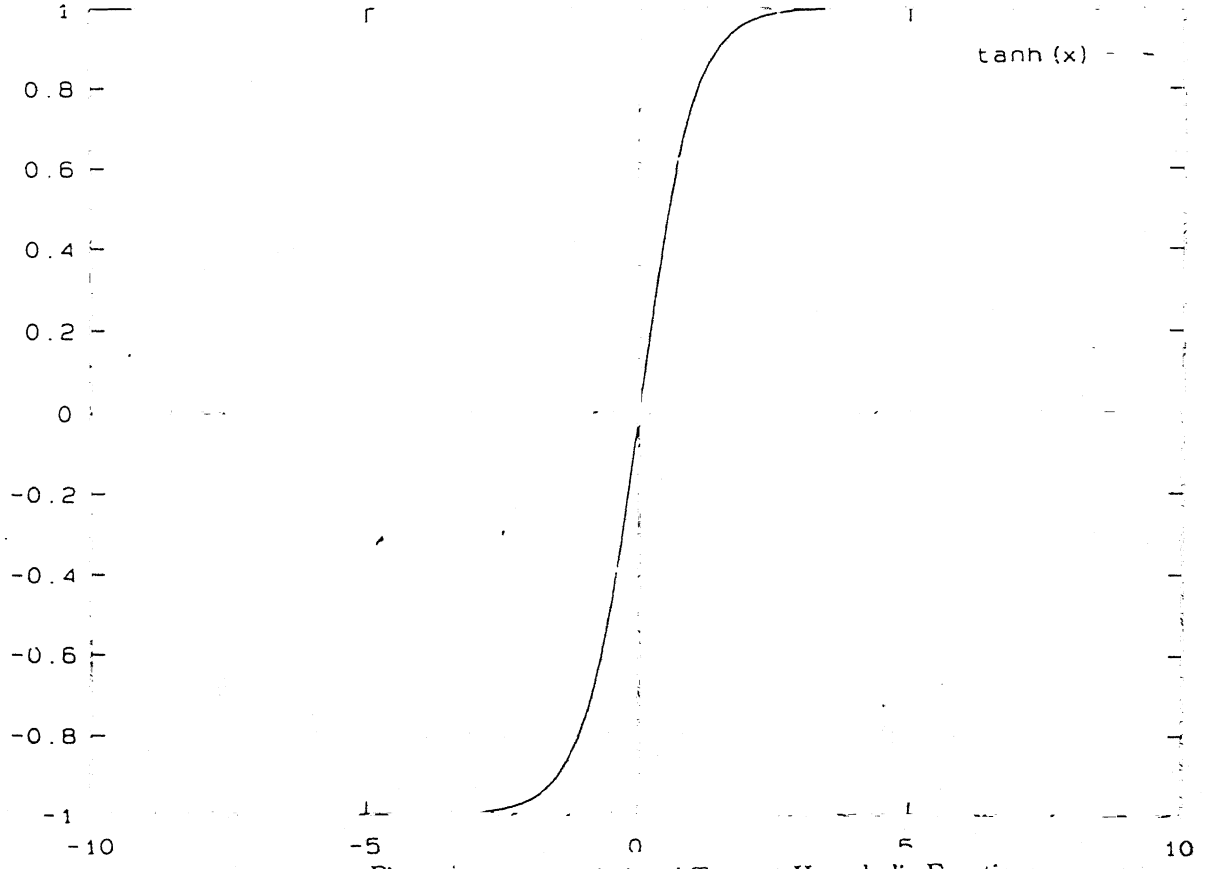


Figure 2.1 Sigmodial and Tangent-Hyperbolic Functions

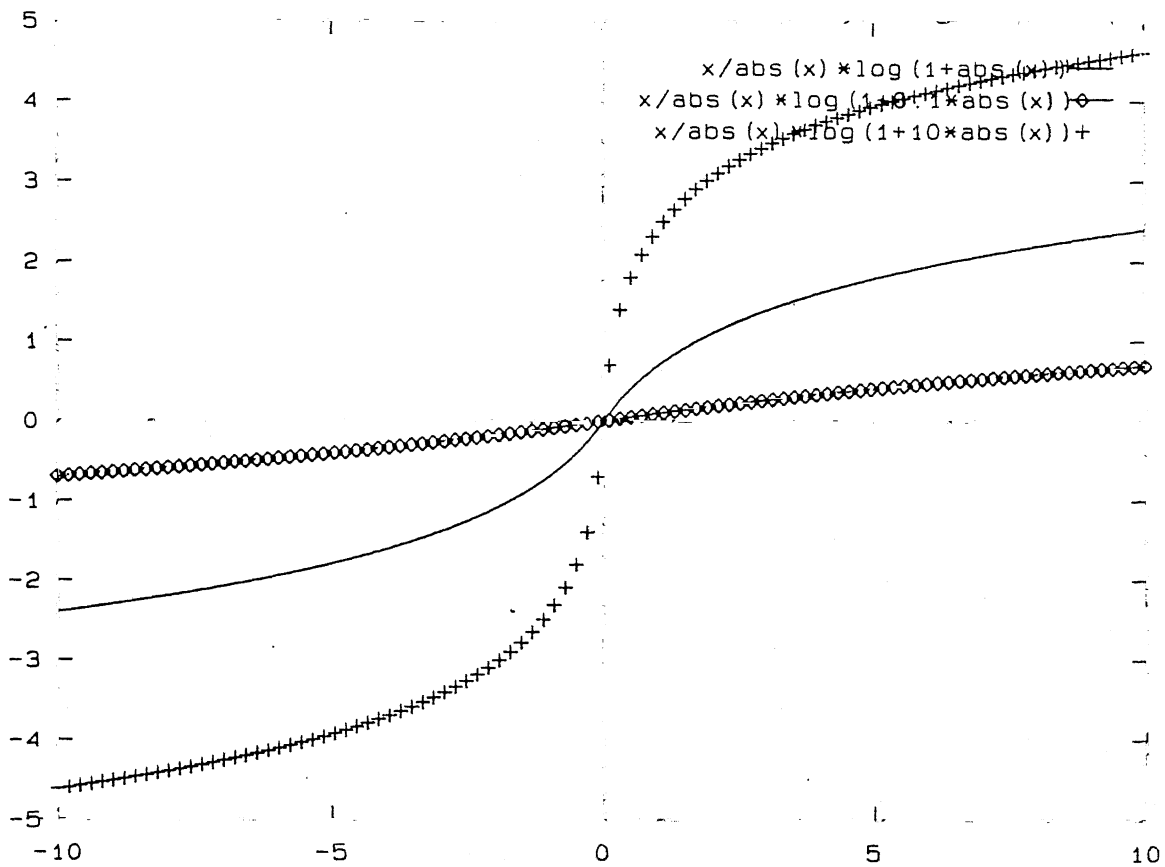


Figure 2.2: The Logarithmoid Activation Function

input signals and the selection of limit between which random weights are generated. For example, if the scaling of input is carried out between 0.1 and 0.9 then random weights are preferred between - 0.5 and + 0.5 for standard sigmodial function. A major drawback of these activation functions is it's saturation which can cause significant errors when the output has no upper bounds.

2.1.2 Selection of Topology

Topology of ANN deals with a) *number of layers neurons in each layers* and b) *their interconnections.* Too few hidden neurons hinder the learning process, while too many occasionally degrade the ANN generalization capability. Clear cut guideline are not available in the literature for deciding the topology of ANN.

One rough guideline for choosing the number of hidden neurons is the *geometric pyramid rule* . It states that, for many practical networks, the number of neurons follow in successive layer a pyramidal shape, decreasing from the input toward the output. The optimal number of hidden neurons can be determined by the **brute force method.** As the number of neurons is fixed in input and output layer so first ANN are trained and tested , starting with a small number of hidden neurons. The number of the neurons is then gradually increased in stages until the error is acceptably small. The **Brute force** method as shown in Figure 2.3 is time consuming but reliable.

Time complexity of topology decision can be reduced by pruning of ANN. This can be done by training of ANN of larger size. due to larger size ANN has less generalization capacity. Afterwards some parts of ANN is pruned to improve generalization and sensitiveness. Different algorithms are described [30] in literature for this purpose.

2.1.3 Error minimization procedures

When ANN are trained weights are modified in order to minimize the error on the training patterns. Different functions can be used to calculate the error. The most common being

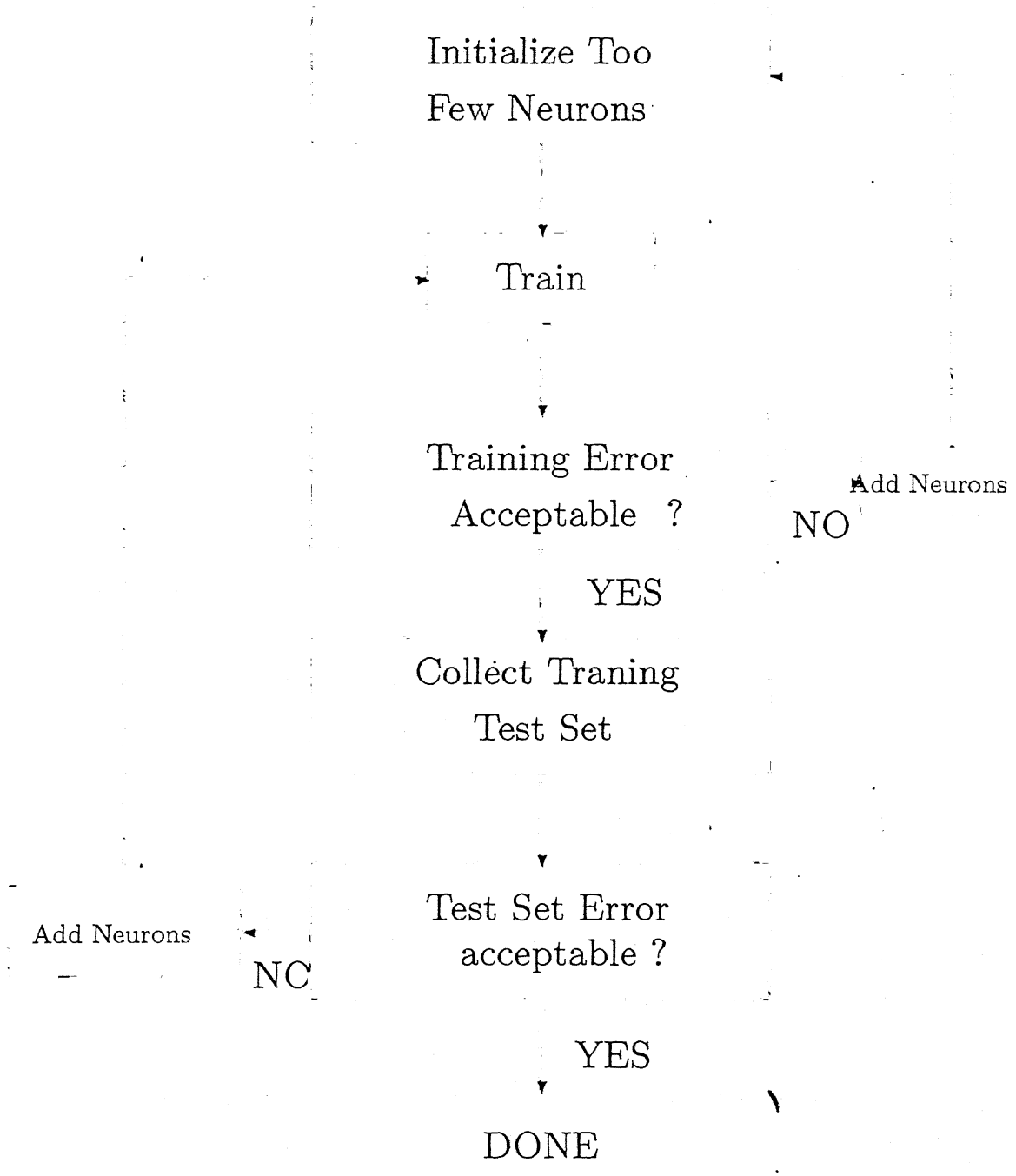


Figure 2.3: Method for Deciding the Topology of ANN

Root Mean Square (**RMS**) function and is given by :

$$Error = \frac{1}{2}\sqrt{\sum (Auctual - Predicted)^2} \tag{2.1}$$

Equation 2.1 can be generalized as :

$$Error = \frac{1}{2}\sqrt[k]{\sum (Auctual - Predicted)^{2k}} \tag{2.2}$$

Where k = Order of Norm

One can also use an error function [25] such as

$$Error = \sum Auctual * \log(Predicted) + (1 - Auctual)\log(1 - Predicted) \tag{2.3}$$

These error function may increase the learning rate of ANN and it's generalization capability.

Generally calculus based techniques are used to minimize error. These methods have poor convergence, as they can be trapped in a local minima, a drawback, which can be eliminated by using evolutionary techniques such as Simulated Annealing [26] and Genetic Algorithms [27] . Unfortunately these techniques are not applicable for fast training.

Different strategies are used for updating the weights of ANN. In one approach, the weights are updated at every cycle of the entire set of training patterns presentations. This process is called the batch or periodic updating. In the second approach, The networks are updated continuously after each training pattern is presented. This approach is called on line or continuous updating. In another approach frequency of a given training samples for updating the weights are also depends upon it's miss classification (error).

In fact learning can be divided in two groups Passive and Active. In Passive Learning all the available examples are fed to ANN for learning, but in the case of Active Learning is not so. In this case ANN is trained with pattern corresponding to maximum error. After the training ANN is tested on reset of the examples and sample giving the worst performance is also added for training. This process is repeated until the ANN perform well on rest of data set. Thus in this case cross validation is done automatically.

Training of ANN is influenced by selection of different parameters. Learning and momentum rate influences the speed of learning in gradient decent technique. Increasing the learning rate may improve the speed the learning but it can also cause oscillations. Even scalling parameters of sigmodial function (well known as *gain*) also influence learning behaviour [28].

In genetic algorithms related techniques the search space can be increased by increasing the population size at the cost of slow learning. Similarly type of phenomena is observed in the case of *simulates annealing* , type of membership functions and different fuzzy operators with *fuzzy networks* and thus attention should be paid in order to decide not only its minimization method but also the above mentioned learning parameters.

2.1.4 Selection of training pattern and preprocessing

Selection of the learning data is very importancet [29] for building ANN. Althöugh preprocessing is not necessary but it pay back handsomely in improving performance and reducing training time. Preprocessing includes scaling, normalization, and noise-reduction.

If the learning data contains linguistic variable like small, very small, large. then data has to be digitized before feeding to an ANN. Different transformations such as linear and log scaling of learning data are also required for good perdiction. If the distribution of a variables is *unusual*, it may be more difficult for ANN to learn to use it, even if the variable is linearly scaled to a reasonable range. This is because the information content in the variable is too distorted. Small but importance variation of variable may be compressed into a relatively narrow area, while other variations are spread out in a wider range than it's important justifies. In this case nonlinear transformation should be adopted. Examination of distribution of variables before and after the transformation must be done. For achieving excellent results preprocessing of learning data is recommended.

2.1.5 Termination Criteria for learning

The process of adjusting weights are generally repeated until the termination condition is achieved. So in practice one has to specify termination conditions. The learning process can be terminated when

- the error goes below a specific value : Implies terminate the learning when *error* is below a specific value.
- the magnitude of gradients reached below a certain value : Do the learning untill the magnitude becomes very small.
- specific number of iteration is complete. : Stop the learning when specific number of iteration is complete.

One can also terminate the learning process by cross validation technique. In this technique data are divided into a training set and validation set. The Training set is used to modify the weights, the validation set is used to estimate the generalization ability.Training is stopped when the error on validation set begins to rise. This technique, may not be practical when only small amount of data is available.

The first three criteria are sensitive to the choice of parameters and may lead to poor results if parameters are not properly selected. The cross validation does not have this drawback. It can also avoid over fitting of the data as shown in Figure 2.4 and which actually improve the generalization performance of the networks. However this technique is time intensive.

2.2 Testing Phase Issues

The performance of ANN on testing data represents its generalization ability. In actual practice the total data is divided into two groups. One is used for learning and the other for testing. In fact a generalized neural networks will perform well for both learning and testing data. Testing must be done for interpolation as well as extrapolation (although it

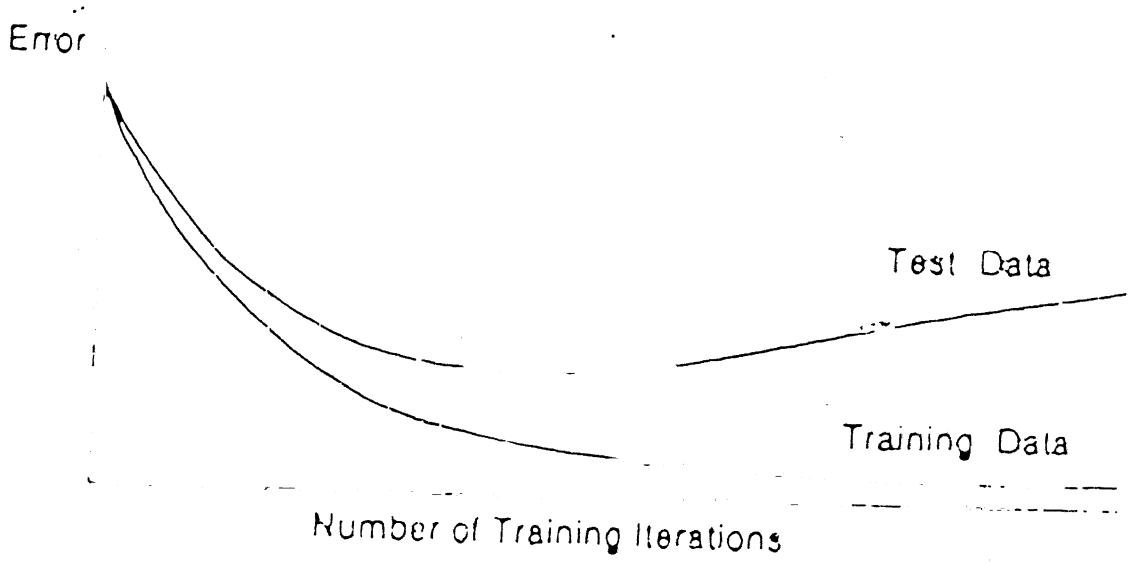


Figure 2.4: An Example Of Over Fitting of Data

is found that the performance ANN is poor for extrapolation). At the time of testing some of the weights ANN are removed to check the stability. If the performance of ANN does not decline then it implies that the ANN will be stable in the case of small hardware / software failure. Testing of ANN only with testing data is not sufficient. Hence small amount of noise is also added to the input to check the stability of the ANN.

2.3 Objective

One of the major objective of this Thesis is to study the some of the mentioned issues and study the *integration* of different AI paradigms. To fulfill this requirement *a simulator*, which can handle different issues, was needed. Now large number of simulators are available for ANN (**GENESIS,SNN,Neural Shell, The Brain**). In these simulators very little attention is paid to integrate Genetic Algorithms, Fuzzy Logic with ANN. So in order to fulfill the requirement a simulator **ANNS** was developed which covers different ascepts of ANN.

Chapter 3

ANNS The Neural Network Simulator – an introduction

3.1 Introduction

This chapter is an overview of the new simulator of ANN, **ANNS** that has been designed to integrate different variation of ANN and to overcome the bottlenecks discussed in *chapter* 2. This ANN simulator is *problem independent* and hence can be applied to wide variety of problems. It also provides **SGA** (*Simple Genetic Algorithms*), an optimization tool independently.

This chapter begins with describing the goal and advantages of the simulator in broad terms. Then in *section* 3 describes the basic concepts that are used in building the simulator. Section 4 describes the current status of work.

3.2 Simulator In General

Significant progress has been made on feedforward ANN and related issues in past few years [18]. Numerous extensions to the basic backpropagation model given by Rumelhart et.al. [35] have emerged. Most of these have been designed to overcome some of the inherent

limitations [chapter 2]. Extensions to the basic Rumelhart model have generated interest in new classes of networks which had evolved from integration of different AI techniques such as fuzzy, genetic with ANN and are capable of performing wide variety of tasks including pattern recognition, nonlinear prediction and system modeling.

3.3 Basic Principle of Design

There are the following three purpose of development of ANNS. 1) to overcome some of the bottlenecks discussed in *chapter* 2, 2) to integrate ANN with different AI techniques. and 3) to test the simulator on some of the problems. So this chapter contains different modification and integration of ANN with different AI technique with ANN.

3.3.1 Back Propagation and Its Modification

The most widely known and applied ANN is *Backpropagation* given by Rumelhart. Sometime this original model is also referred as Standard Backpropagation (SBP). This model is based on the error minimization technique called gradient decent. By using the above technique the correction of weights in different layers are done by a chain rule popularly known as *Generalized Delta Rule* [4].

Supported algorithms are widely available in literature [9], [10], [11]. Although this algorithms have been widely applied for different problems but it suffers from serious drawbacks. Some of the more important among these are slow speed, trapping in local minima, misdirection of decent, oscillations [9].

3.3.2 Change in Initial Guess for Gradient Methods

Attempt have been made to overcome these drawbacks. The problem of getting trapped in the local minima may be eliminated by selecting different random generated weights (as the success of gradient search method depends upon initial guess) and proper scaling of input attributes. Provision has been made a) to give different number seed for generating random weights and b) different type of linear scaling has been performed in this stimulator.

3.3.3 Modification of Activation Function

a. Modification of Sigmoid Function

Generally sigmoid function is used as an non linear filter. Which can be given as

$$F(x) = \frac{1}{1 + \exp(-x)} \tag{3.1}$$

According to Caudil [36] if we use slightly modified sigmoid function as

$$F(x) = -0.5 + \frac{1}{1 + \exp(-x/t)} \tag{3.2}$$

then convergence time may be reduced by half compared with the sigmoid function which has 0 to 1 (Equation 3.2). Hence provision has been made to capture this modification in the simulator by providing the nonlinear filter as :

$$F(x) = -r + \frac{k}{1 + \exp(-x/t)} \tag{3.3}$$

Again choosing this activation function forces to scale the output attributes between $-r$ and $-r+1r$ (as the output of the above function varies from $-r$ to $-r+1$). This simulator also provide different type of output scaling to capture this modification.[Variation No. 1.1].

b. Change of Activation Function (Logrithmoid Function)

The sigmoid function is used as a filter in the above algorithm has a drawback that it restricts the output between certain range such as $-r$ to $-r+1$ and forced to scale the output between these limits. Hence this function is not suitable for those process where output are unconstrained. The Logrithmod function's is monotonically increasing, continuous and unbounded. It's shape is similar to tanhyperbolic. This symmetric logrithmoid function provides an alternative of sigmoid function. Claims have been made that it may work better than sigmoid functions [21] in performance as well as in convergence. This function is shown in Figure 2.2. The derrivative of this function is as follows $C * \exp(-\text{ mod } f(x))$

As the derivative contains the exponential term which implies that giving high value to learning and momentum rate can causes network paralysis. Thus learning and momentum terms are kept low in this variation [Variation 1.2].

3.3.4 Change of Error Function

Generally error during learning is defined as a function of desired output and the network's output. Error is minimized by using training data set by modifying weights. Generally RMS method is used to calculate the error 2.1.3. Thus derivative of error is calculated with respect to weights. The modification of the weights are performed proportional to the derivatives. Using RMS error function the term $[output * (1output)]$ 2.2 is obtained in the gradient. Thus if the output is either near to zero or one the value of gradient will be small and the correction of weights should be less making learning process very slow. In most of the cases the output is scaled between 0 and 1. Thus there is a high probability of slow learning when output may go either near to 0 or 1. This problem can be eliminated by using a different error function as given below

$$Error = \sum Auctual * \log(Predicted) + (1 - Auctual) \log(1 - Predicted) \tag{3.4}$$

3.3.5 Removal of Influence of Pattern Presentation

ANN is trained with training data set. The ANN weights are updated continuously after each pattern is presented in the case of online training. Hence the sequence of the pattern presentation may influence the error minimization process and thus the weight correction. This influence may cause loss in generality of ANN. The influence of the pattern sequence may be eliminated by shuffling the data set randomly after each iteration. The random shuffling of training is also incorporated in the simulator.[variation 1.4]. Block training is also used to remove the influence of pattern presentation. However this has not been dealt in simulator.

3.3.6 Decision of Frequency of a Particular Data Set

Some data sets produces more error than other in the process of training . This may be due to two reasons.

First the information available in that data set is not gathered by ANN. In this case training time as well the performance may be improved by increasing the frequency of misclassified data set. In this variation the data sets are fed according to their error values. The data sets which have large error has more frequency than the data set having less error. The number of presentation is calculated as

$$Number = \frac{E_k * P}{E} \tag{3.5}$$

where

$E_k = Error in K^{th} pattern$

$P = Number of patterns$

$E = Total Error of all Patterns$

The second reason may due to the fact that the data which is not able to classified may be a bad data (that data which which not represent the actual process) . Thus care should be taken in selection of training data as well as the learning and momentum rate for this variation [variation 1.5]. Giving high learning and momentum in this process give more weight to bad data which may cause oscillation and get converted to local minima.

3.3.7 Radial Basis Function

The above algorithms are basically improvement of standard backpropagation algorithms. All of them are based on supervised learning. One of the disadvantage, *large trainig time* can be reduced by using Radial Basis Function networks (RBF), which has supervise plus unsupervised learning. This networks uses Gaussian function as a nonlinear filter. This is called Radial Basis Function . Networks consist of only three layers input, one hidden and an output layer. The input to hidden layer uses unsupervised learning while hidden to

output layer uses supervised learning. The unsupervised learning is a clustering algorithm. This simulator uses one of the popular clustering algorithm, K-mean [18].

The details of the algorithm is provided in appendix A. The working of this network can be explained in the following way.

Let n be the number of training data set. Clustering algorithm clusters them into m group (obviously $m < n$). With the help of cluster σ is calculated. Thus the output of the hidden layers are determined. consequently corresponding weights can be calculated in order to get the output. Thus this network uses batch learning.

The RBF networks can be used for both classification and function approximation. Using a set of non linear basis functions, the RBF network is capable of performing any arbitrary mapping. The main difference between the RBF network and the backpropagation network is in their basis functions. The Radial Basis Function in the former networks covers only small regions, whereas the sigmoid function assumes nonzero values over an infinitely large region of the input space. The classification problem can be handled very well by RBF than backpropagation. The algorithm of RBF networks is given in Appendix A.

3.4 Integration With Different AI Techniques

3.4.1 Integration with Genetic Algorithms(GA's)

As stated earlier,2.3 the basic objective of this simulator is to integrate the different AI technique with ANN.In the first stage ANN is integrated with GA's [variation 3]. Genetic Algorithms is an evolutionary technique use to find global optima [16]. In this case GA's are used to obtain weights corresponding to a global minima to and hence may provide better learning than gradient methods.

Backpropagation 2 starts with random sets of weights and uses gradient method to upgrade the weights in order to minimize the error. Success of this backpropagation depends upon initial guess and nature of the error surface. In the case of backpropagation the error

surfaces form complex hyperplane which may consists of many local minima and maxima. Presence of local minima increases the failer with calculus based search techniques.

Genetic algorithms [2], an evolutionary technique, is helpful in finding out the global optima. GA's works with three basic operators called *Selection (reproduction), Crossover and Mutation* evolved from *natural adaptation* process [16].

The objective function which has to be optimized by Genetic Algorithms is given by

$$Fitness = \left(\frac{1}{1+E} \right)^{\gamma} \tag{3.6}$$

Where E is the RMS error calculated from all the available training sample by forward processing of SBP. were

$\gamma = fitnessparameter(> 1)$

GA's used to maximize the above function by adjusting the weights to maximize the objective function which actually minimize the learning error.

GA's is based on selection of genetic parameters such as population size, number of generation, selection procedure, crossover procedure, mutation procedure crossover probability, mutation probability, length of chromosome (precision for each weights) [3]. Excellent results may be achieved by choosing proper GA's parameter. This simulator provides two type of selection procedure (roulette wheel and tournament selection) and crossover (single point and multiple point) operators [2] - [3]. Technically how the GA's is implemented is given in Appendix B (GA And ANN).

The solution from GA's may be used as an initial guess for backpropagation. This provision has been made to continue standard B.P. in order to achieve relatively better result variation[3].

3.4.2 Integration of ANN with Fuzzy Logic

The fuzzy neural networks are well known for their ability to handle the fuzzy (inexact) nature of interface involving symbols (symbolic inference). In fuzzy logic, a linguistic variable like size can have several linguistic values such *small, medium* and *large*. Each

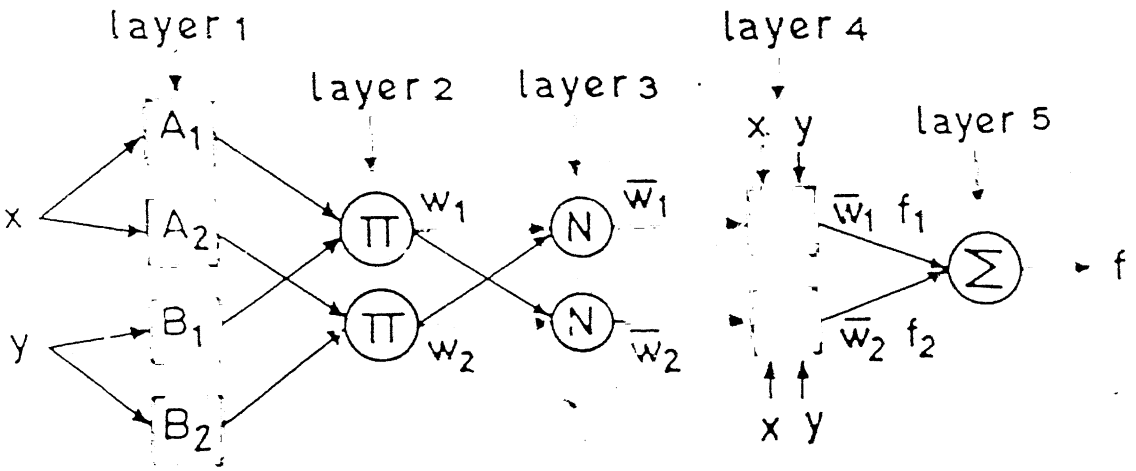
linguistic variable can be viewed as a fuzzy set associated with a membership function, which can be triangular, bell-shaped or of another form. The membership value represents the degree of belongingness to a particular class or group.

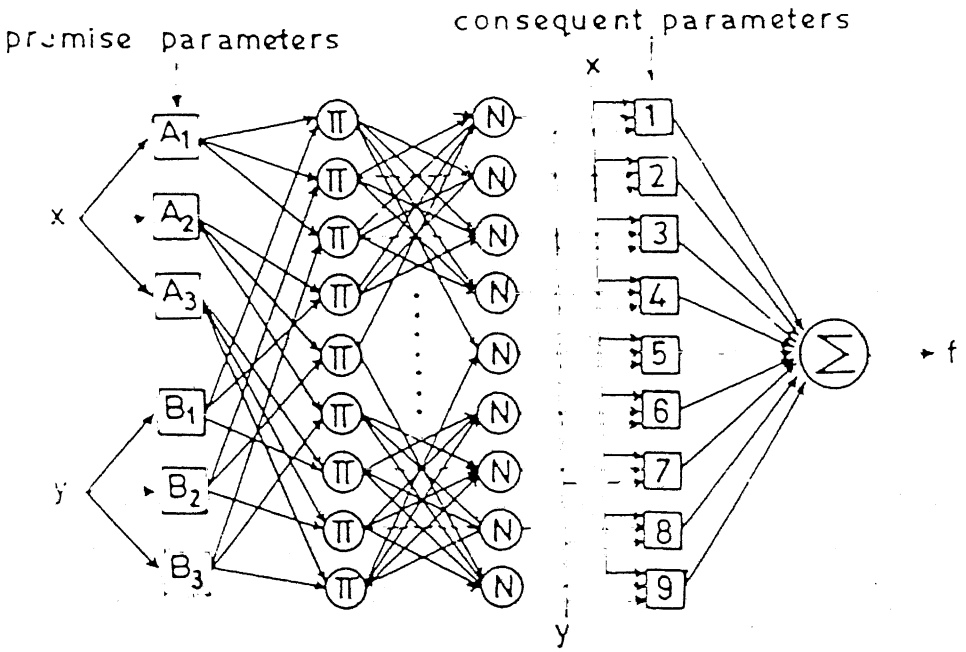
Fuzzy neural network works in the following way. It accepts crisp input value and fuzzify according to it's membership function. Membership function is modified by database and the rulebase of the problem. The second task is to make different fuzzy rule base. These rule base generate fuzzy consequent, which has to be defuzzify in order to obtain crisp output. Thus Fuzzy Neural Network is made up of at least five layers as

- input layer
- input fuzzy layer
- conjunction layer
- output fuzzy layer
- output layer

In order to obtain correct conclusion one has to adjust different parameters associated with these layers. These layers are made up of nodes which may or may not have parameters or rules to adjust. One of such network called ANFIS [variation 4] has been incorporated accumulated in this simulator in order to integrate fuzzy decision making with artificial neural networks [15]. It has two types of rule bases associated with it.[variation 4.1 and 4.2].

The link in this network only indicates the flow direction of the signal between these nodes (Figure 3.1). The node can be divided into two groups fixed node and adaptive node. The adaptive node has parameters which are adjusted in the process of learning while fixed nodes have no parameters (in figure the adaptive node is indicated by square node and fixed node by circular one). In order to achieve desired input output mapping parameters associated to adaptive nodes are updated according to the given training data





EQUIVALENT FNN FOR TWO INPUTS AND NINE RULES

Figure 3.1: ANNFS - The Fuzzy Neural Networks [15]

and a gradient based learning procedure. Even the step size is governed by two linguistic rule [15]. The learning process in this case is batch learning.

In conventional fuzzy interface system, the number of rules are decided by an expert who is familiar with the system to be modeled. In this simulation, however no expert is available and the membership function (MF's) assigned to each input variable is chosen empirically, i.e., by examining the desired input-output data and/or by trial and error. This situation is similar as that of ANN; as there is no simpler ways to determine in advance the minimal number of hidden nodes necessary to achieve a desired performance level.

3.4.3 Integration of ANN with Genetic algorithm and Fuzzy logic

Membership functions play very important role in fuzzy decision making. Thus attention are paid in proper selection of membership function which may lead to excellent results in fuzzy decision making. The same facts are also true in the case of fuzzy neural networks. In the case of fuzzy neural networks the membership function is defined as

Whose shape can be varied by changing the value of a,b,c, the premise parameter. In the variation 4 the final value of premise parameter are determined by gradient decent method which itself suffers from a large number of limitations as mentioned above.

This indicates that one could go for better method for determination of these premise parameters. GA's is one of the option among them which may help in finding better values of these parameters.

This variation (Variation 5) is actually modification of variation 4 in which premise parameters are selected by using GA's. Again the the objective function in this case is same as defined in equation 3.6. The values of premise parameters are adjusted with the help of GA's to optimize the objective function which actually minimize the learning error.

Again this variation provides selection of different GA's parameter such as population size, number generation, crossover probability, mutation probability. Two different selection scheme roulette wheel and tournament selection and two type of crossover single point and multiple point are also incorprated in this variation.

3.4.4 User Interface For Simulator

The simulator **ANNS** is developed from user point view. Each input followed by proper menu. Proper explanation is provided in all type of ANN. This simulator also contains *".default"* file which is generated by user during the process of providing the input to simulator. Thus use can use the simulator with different parameters by changing only one entry in *".default"* file. The input data file contains input and consequent output in sequence.

All the modules of the simulator are independent as well as hardware independent. This can be used on any unix plateform which has standard "C" compiler(ANSI C). No global variable is used in this simulator. So this simulator can also provide good shell for future development of different modules related to ANN.

Chapter 4

Results And Discussion

4.1 Introduction

Different variations of **ANNS**, the neural network simulator, is tested on two problems : short term load forecasting (*Test problem No 1*) and determination of oxygen mass transfer in *air agitated* Pachuca Tank (*Test problem No 2*). This chapter contains description of the test problems and their result obtained from **ANNS**.

4.2 Short Term Load Forecasting – *Test Problem No 1*

Short term load forecasting plays an important role in day to day operation and scheduling of power systems. Accurate forecasting of electric load is required to balance the supply and demand of electricity. Again it helps in energy transaction, system security analysis and optimum operation planning of power generation facilities.

Different techniques such as time series method and parameter estimation are generally used for load forecasting. These methods fail to give reasonable accuracy because of their inherent limitation such as unstability and functional dependence [38]. Again different techniques such as artificial neural network, mainly backpropagation, fuzzy regression are used

to estimate the load [38]. All these techniques individually suffer from a lot of limitations mentioned in *chapter 1.*

The basic objective of development of *ANNS* was to overcome some of the inherent limitations of ANN as well as to make it more powerful by integrating it with *fuzzy logic* and *genetic algorithms.* Different variations of *ANNS* have been tested for the load forecasting.

4.2.1 Problem Formulation

The electric load of a power station depends on many factors such as day of the week, speed of the wind. Since the day of the week and load of the previous hour (*past history*) are the most important variables, they are used as an inputs to the ANN. The load of the three previous hours are considered as past history for past history.

The data set of all mondays (the day selected for forecasting) is divided in two groups - 1. training data set *consist of data of first seven Mondays* and 2. testing data set *the data of the* 8^{th} *Monday.* As the name implies training of ANN is done by training data set while testing data set is used to test the ability of ANN to forecast the load. Results of different variations of *ANNS* are discussed below.

4.2.2 Variation 1 *BP And It's Modification*

BP and its four modification are run for 10,000 epochs for six different network topologies. In all the cases the starting weights are generated between small ranges typically between -0.5 and 0.5, both training and testing data are scaled between 0.1 and 0.9, learning rate and momentum factor are set equal to 0.7 and 0.5 respectively except in variation 1.3 and 1.5 where both learning rate and momentum factor are kept vary small [Chapter 3].

Table given in page 37 to 46 presents the results of BP (Variation 1) and its modification. From the predicted results it can be concluded that for bigger network, the performance in learning data set improved but the performance of testing data sets detoriates. For example in the case of standard backpropagation variation the net 3:9:6:1 (large

net in that variation) gives low standard deviation for learning data compared to testing data set (page 38).

The logarithmic error function (Variation 1) is used to improve the convergence of ANN with all the toplogy (in general). Hence comparison of the results with SBP (page 37 and 38) with this variation (page 39 and 40) shows that this variation does not improve the convergence.

Different nonlinear filter (logarithmoid function variation 1.3) is also included as an alternative to sigmoidal function. In this variation learning rate and momentum factor are kept very low (0.05 and 0.05 respectively) [Chapter 3]. The results indicate that it's performance and convergence are comparable with variation 1.1 (page 37).

When patterns are presented in random order in each epoch (Variation 1.4) the generalization capability of the network improves in this test problem. For example in the network 3:9:6:1, standard deviation of testing data is 100.212 for standard backpropagation while randomize shuffling gives a standard deviation of 90.378 (Table in page 43 to 44).

In variation 1.5 (discussed in Chapter 3), pattern are presented in proportion to their error values at each epoch. Using this variation, the error does not decrease as rapid as expected. The Learning and momentum rate is kept vary low [Chapter 3]. The results indicate that the final error of standard backpropagation and this variation are greater than SBP (Table in page 45 and 46) in this variation.

4.2.3 Variation 2 *RBF network*

The results of RBF network (Variation 2) are shown in table in page 47 and 48 for different network topologies. The number of nodes in the middle layer represent the number of clustering center and this number was varied for different topologies. Six different topologies (6 different nodes in hidden layer) have been tried. It is clear from the table that as the number of hidden nodes are increased, the learning capability of the network increases. The network 3:30:1 gives the best generalized results compared to all other network topologies (standard deviation 135.34 for training and 107.46 for testing) of RBF Networks.

4.2.4 Variation 3 *Integration with GA's*

This variation is integration of genetic algorithm with ANN (Chapter 3). Page 49 and 50 contains the results of six different topology of this variation.

Different parameters of GA's used in this problem as

- Population size = 1000
- Number of generation = 500
- Probability of crossover = 0.9
- Number of crossover site =4
- Mutation probability = 0.085
- Binary tournament selection is opted for selection scheme.
- Search space for the weight is taken from -20 to +20 that is $w_{min} = -20$ and $w_{max} = 20$.

After running GA's for 500 generations, the best ever chromosome is decoded in to real weights. These weights are selected as initial solution to the backpropagation algorithm and SBP is run for 1000 iterations. Learning rate and momentum factor is chosen as 0.7 and 0.5 respectively. $Bestever^{ingen}$ shows the generation in which best chromosome is found by GA's. It can be deduced that learning stops at early stage. For example the $Bestever_{ingen}$ is found in 410, 113, 56 ... generation respectively (page 49 and 50).

The learning error (after 500 generation of GA and 1000 runs for BP) is higher. For example the network 3:6:6:1, the RMS error is 0.0434 (page 50) while GA's based gives 0.06287.

Since the variation has poor learning performance, hence the prediction for unseen data is also inferior than SBP.

Training time is also very high in this variation. For example, with these parameters forward processing is done for $500 * 1000 = 5,00,000 + 1000(\ SBP\ ) = 5,01,000$ times for finding optimal weights from GA's.

4.2.5 Variation 4 *Integration with Fuzzy Logic*

Variation 4.1

Result of this variation are given in page 51 and 52. Different network are trained by varying the number of membership function associated with each input. The number of iteration is selected 1,000 and step size =0.5.

Some of the topologies (network with different number of membership functions associated with each input) showed better performance than all previous networks. In the case of four membership function associated with each input RMS error is 2.69 better than BP and RBF network. This variation indicates that if the number of membership associated with each input is increased then learning performance will increase but it will degrade the generalization capability. Results [page 52] indicates that learning performance with eight membership functions associated with each input is excellent but it fails for testing data set.

Variation 4.2

This variation is only due to increase in the number of rules. The results due to two membership functions are excellent but again the network looses its generalization capability when number of membership function associated with each input becomes 3 (page 53).

4.2.6 Variation 5 *Integration with Fuzzy Logic And GA's*

This variation is an integration of fuzzy and genetic algorithm with neural network. GA is applied for selection of membership function. It uses binary tournament selection schemes. Multiple point crossover (4 sites) is used with a probability of 0.9. The mutation probability is kept as 0.05. The population size is kept at 500 and termination criteria is 100 generations.

The results of this variation are given in at page 54 and 56. The results of 4 membership function associated with each input is *best among others networkas*. The results (54 to 55) also indicate that by increasing number of membership function associated with each

input, the learning performance will increase but will degrade the testing performance. For example, with 6 rules the RMS error value of learning data is 3.11 and 4.78 RMS error with testing data while with five membership function it is 3.32 and 3.47 RMS error.

On the other hand variation 2 showed better result for two membership functions associated with each input (page 54). It produces RMS error of 2.98 for testing data set. The performance with three membership function is poor for this test problem.

4.3 Oxygen Mass Transfer In Air agitated Pachuca Tank – *Test Problem No 2*

Pachuca tanks are used extensively for carbonate leaching of uranium and cyanide leaching of gold in which oxygen mass transfer plays a very important role. Experiments have been carried out in laboratory scale Pachuca Tank to clarify the effect of design and operating system on oxygen mass transfer[37]. The model of Pachuca tank used in this study is shown in page 75.

The laboratory scale Pachuca Tank was designed such that certain important design and operating parameters, namely height to diameter ratio of the tank, draft tube and to tank diameter, cone angle and superficial gas velocity (Vg), where in the same range as these are used in industry (Table in page 58). Experiments have been carried out in four tanks of diameter of 0.15, 0.25, 0.36 and 0.56 m. Each set consists of variables as diameter of the tank, height and diameter ratio of the tank, diameter ratio of draft tube and tank superficial gas velocity as the input and volumetric mass transfer as an output.

In this test problem mass transfer rate is modeled with the help of different variation of AAN using the simulator *ANNS*.

The experimental data are divided into two groups called training and testing sets. Data from tank one, two and three are taken as training data set. Tank four data is selected as testing data set.

Results obtained by different variations are discussed below.

4.3.1 Variation 1 *BP And It's Modification*

The SBP and it's modification for six different topology are tried. These are given in page 59 to 68. Again the scaling of data sets is done from 0.1 to 0.9 by using the maximum and minum table given in in page 58.

Learning rate and momentum factor are kept 0.5 in most of the cases. The values of logistic is kept at 0.7 in all the BP model except variation 1.3 where it is kept as 1. The weights are generated between -0.5 and 0.5. Training is done for 1000 iterations.

Result shown in different tables indicate that in this case variation 1.3 gives better performance than other networks. The best results are obtained from net 4:1:1 which gives 14.4 % RMS error in prediction and 15.72 % RMS in learning data set (Better performance better than earlier model). However application of Variation 1.2 does not improve the performance significantly.

4.3.2 Variation 2 *RBF network*

Result of RBF Networks with six different topology are shown in table in page 69 to 70. The best performance (RMS Error 27% approx with testing) obtained in this case is with 4:6:1 networks. Thus overall performance of RBF networks is poorer than backpropagation.

4.3.3 Variation 3 *Integration with GA's*

Page 71 and 72 shows results of different variation of GA's based neural network. The different parameters of GA's selected for this prediction is same as there in given in test problem number 1. In this case again learning by GA's stopped in early stages as the $Bestever_{ingen}$ is achieved in 56, 32, 300, 261, 59,... generations (page 71).

This learning performance is poor in this case which gives rise the poor performance for both learning as well as testing data sets. Learning error after 1000 iteration of gradient search methods does not reduces to the level of SBP

4.3.4 Variation 4 *Integration with Fuzzy Logic*

The results of variation 4.1 with different membership functions are shown in page 73. The parameter selected same as those in test problem number 1. The performance of this networks is poorer. Increasing the membership function gives rise to over learning and hence performance with testing data set is very poor.

This variation uses batch learning. In the case of variation 4.2 if two membership function are attached to each input then second layer has 2 * inp number of nodes, where as inp is the number of input of ANN. Here the number of consequent parameter will be 2 * (inp + 1). For this problem therefore at least 80 data set are required for learning. Due to less number of data this variation is not tried for this problem.

4.3.5 Variation 5 *Integration with Fuzzy Logic And GA's*

In this variation the membership function is selected by GA's – a modification of variation 4. The GA's parameters used here are same as there in test problem number 1. Variation 5 suffers from over learning and unable to produce the desired result (page 74).

4.4 Results

In the case of BP and it's modification the best result obtained in two different problem are different. The power prediction can be done better with random data shuffling while use of logarithmoid activation function will produce better result in the Pachuca Tank problem with BP.

Best results obtained from different algorithm are shown in page and for power forecasting and Pachuca tank problem. As stated earlier *integration of ANN with genetic algorithms and fuzzy logic* produces best result in the power perdition problem. On the other hand logarithmoid function process best results in Pachuca Tank problem. Comparing the best result obtained by ANN and the results obtained by Roy [37] by multivariable

regression showed in graph (page 80) indicates that ANN is better option for Pachuca Tank problem.

Again results produced in test problem 1 is better than test problem 2 in the case of RBF network. Test problem No. 1 is a problem which varies in space as well in time while in test problem No. 2 the mass transfer rate varies linearly with V(g) [37] . As problems of higher dimension can be easily handled by RBF networks which may be the reason of better performance of test problem no. 1.

Integration of GA's with ANN does not work well in both the test problem. GA's fixed the search space of weights and which may lead to failure of this algorithm.

The performance of Fuzzy networks is better than BP and it's modification as well as RBF networks. But in the case of test problem no. 2 it may produces problem due to overlearning. The integration with Fuzzy as well as GA's gives best result in the case of test problem no 1.

The best result of these two problems are shown (page 79 and in) for above two test problems.

Learning Rate : 0.70 Momentum : 0.50

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:6:1	3:6:9:1	3:9:6:1
End Error	0.043448	0.0432022	0.042846
$Error_{rms}^{learn}$ %	4.334	4.0824	4.4192
$Correlation^{learn}$	0.9063	0.9106	0.91028
$Deviation_{stand}^{learn}$	114.509	102.494	102.664
$Error_{rms}^{test}$ %	3.2952	3.23048	3.6308
$Correlation^{test}$	0.93945	0.9376	0.9199
$Deviation_{stand}^{test}$	88.31	90.44	100.21

Table 4.1: RESULT BP Variation 1.1 for test problem 1

Learning Rate : 0.70 Momentum : 0.50

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:1	3:9:1	3:12:1
End Error	0.04487	0.044494	0.04692
$Error_{rms}^{learn}$ %	4.3245	4.2157	4.7095
$Correlation^{learn}$	0.90242	0.90553	0.89358
$Deviation_{stand}^{learn}$	106.921	105.43	111.03
$Error_{rms}^{test}$ %	3.4993	3.4658	3.6779
$Correlation^{test}$	0.92662	0.92724	0.92433
$Deviation_{stand}^{test}$	95.8978	95.676	96.8983

Table 4.2. RESULT BP Variation 1.1 for test problem 1

Learning Rate : 0.30 Momentum : 0.50

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:1	3:9:1	3:12:1
End Error	0.05090	0.04822	0.04774
$Error_{rms}^{learn}$ %	4.683	4.351	4.175
$Correlation^{learn}$	0.8924	0.900	0.9021
$Deviation_{stand}^{learn}$	112.19	108.78	108.07
$Error_{rms}^{test}$ %	3.481	3.449	3.503
$Correlation^{test}$	0.9297	0.9245	0.9198
$Deviation_{stand}^{test}$	93.78	97.46	100.48

Table 4.3: BP Variation 1.2 for test problem 1

Learning Rate : 0.30 Momentum : 0.50

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:6:1	3:6:9:1	3:9:6:1
End Error	0.04254	0.04406	0.04435
$Error_{rms}^{learn}$ %	4.308	4.276	4.277
$Correlation^{learn}$	0.9120	0.9070	0.9069
$Deviation_{stand}^{learn}$	101.145	104.05	104.153
$Error_{rms}^{test}$ %	3.452	3.722	3.661
$Correlation^{test}$	0.9239	0.9108	0.9135
$Deviation_{stand}^{test}$	98.01	106.85	105.05

Table 4.4: BP Variation 1.2 for test problem 1

Learning Rate : 0.05 Momentum : 0.05

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:1	3:9:1	3:12:1
End Error	0.04419	0.04450	0.04510
$Error_{rms}^{learn}$ %	3.94	4.11	4.28
$Correlation^{learn}$	0.9036	0.9040	0.9021
$Deviation_{stand}^{learn}$	107.01	106.52	107.51
$Error_{rms}^{test}$ %	3.56	3.60	3.69
$Correlation^{test}$	0.9204	0.9201	0.9186
$Deviation_{stand}^{test}$	100.49	100.06	100.65

Table 4.5: BP Variation 1.3 for test problem 1

Learning Rate : 0.05 Momentum : 0.05

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:6:1	3:6:9:1	3:9:6:1
End Error	0.0447	0.04622	0.0451
$Error_{rms}^{learn}$ %	4.38	4.59	4.47
$Correlation^{learn}$	0.9042	0.9009	0.9042
$Deviation_{stand}^{learn}$	105.90	108.37	106.29
$Error_{rms}^{test}$ %	3.62	3.77	3.70
$Correlation^{test}$	0.9265	0.9222	0.9225
$Deviation_{stand}^{test}$	96.10	97.89	98.29

Table 4.6: BP Variation 1.3 for test problem 1

Learning Rate : 0.70 Momentum : 0.50

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:1	3:9:1	3:12:1
End Error	0.04825	0.04551	0.05012
$Error_{rms}^{learn}$ %	4.11	3.85	4.08
$Correlation^{learn}$	0.8900	0.9066	0.9034
$Deviation_{stand}^{learn}$	113.16	105.75	109.49
$Error_{rms}^{test}$ %	4.14	3.46	4.03
$Correlation^{test}$	0.9015	0.9216	0.9230
$Deviation_{stand}^{test}$	112.35	98.11	101.17

Table 4.7: BP Variation 1.4 for test problem 1

Learning Rate : 0.70 Momentum : 0.50

Logistic = 1.00
Number of Iteration = 10000

Net Topology	3:6:6:1	3:6:9:1	3:9:6:1
End Error	0.0455	0.0471	0.0463
$Error_{rms}^{learn}$ %	3.86	4.11	3.93
$Correlation^{learn}$	0.9870	0.9130	0.9075
$Deviation_{stand}^{learn}$	104.19	102.50	104.66
$Error_{rms}^{test}$ %	3.31	3.50	3.34
$Correlation^{test}$	0.9372	0.9381	0.9365
$Deviation_{stand}^{test}$	92.20	90.37	91.86

Table 4.8: BP Variation 1.4 for test problem 1

Learning Rate : 0.05 Momentum : 0.05

Logistic = 0.70
Number of Iteration = 10000

Net Topology	3:6:1	3:9:1	3:12:1
End Error	0.0443	0.0442	0.0440
$Error_{rms}^{learn}$ %	3.94	3.944	3.96
$Correlation^{learn}$	0.9010	0.9013	0.9002
$Deviation_{stand}^{learn}$	109.10	109.23	109.50
$Error_{rms}^{test}$ %	3.79	3.84	3.86
$Correlation^{test}$	0.9119	0.9109	0.9094
$Deviation_{stand}^{test}$	105.94	106.56	107.20

Table 4.9: BP Variation 1.5 for test problem 1

Learning Rate : 0.05 Momentum : 0.05

Logistic = 0.70
Number of Iteration = 10000

Net Topology	3:6:6:1	3:6:9:1	3:9:6:1
End Error	0.0444	0.0446	0.0447
$Error_{rms}^{learn}$ %	3.95	3.95	3.951
$Correlation^{learn}$	0.9015	0.9020	0.9015
$Deviation_{stand}^{learn}$	108.99	108.47	108.19
$Error_{rms}^{test}$ %	3.78	3.70	3.70
$Correlation^{test}$	0.9129	0.9150	0.9153
$Deviation_{stand}^{test}$	105.33	103.98	103.78

Table 4.10: BP Variation 1.5 for test problem 1

Number of Center	10	30	60
Error of RBF	0.820830	0.0686255	0.0580647
$Error_{rms}^{learn}$ %	4.67	3.87	3.74
$Correlation^{learn}$	0.860867	0.9050	0.932977
$Deviation_{stand}^{learn}$	128.516	107.476	90.93
$Error_{rms}^{test}$ %	5.23	5.40	6.79
$Correlation^{test}$	0.84569	0.85674	0.845324
$Deviation_{stand}^{test}$	143.22	135.344	154.624

Table 4.11: RBF network Variation 2 for test problem 1

Number of Center	80	100	120
Error of RBF	0.058064	0.047550	0.03839
$Error_{rms}^{learn}$ %	3.0842	2.740	2.232550
$Correlation^{learn}$	0.943130	0.9555	0.971263
$Deviation_{stand}^{learn}$	84.03	74.49	60.15
$Error_{rms}^{test}$ %	8.76	7.790	5.307
$Correlation^{test}$	0.69234	0.761689	0.6833
$Deviation_{stand}^{test}$	197.616	180.208	195.267

Table 4.12: RBF network Variation 2 for test problem 1

GA's Parameters :

Population Size = 100 Max Generation = 500
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.085
Tournament Selection

Gradient Search Parameters
Learning Rate = 0.70
Momentum Rate = 0.5
Logistic = 1.0 Number of Iteration = 1000

Net Topology	3:6:1	3:9:1	3:12:1
End Error(GA's)	0.05086	0.06072	0.07299
End Error(B.P.)	0.04762	0.05307	0.05669
Best Ever $^{\text{in gen}}$	410	113	56
Error$^{\text{learn}}_{\text{rms}}$ %	4.45	5.98	5.91
Correlation$^{\text{learn}}$	0.89218	0.85560	0.8264
Deviation$^{\text{learn}}_{\text{stand}}$	112.095	125.602	137.22
Error$^{\text{test}}_{\text{rms}}$ %	3.733	4.414	4.271
Correlation$^{\text{test}}$	0.911821	0.9173	0.90169
Deviation$^{\text{test}}_{\text{stand}}$	105.04	100.094	111.755

Table 4.13: Variation 3: Integration with GA's for test problem 1

GA's Parameters :

Population Size = 100 Max Generation = 500
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.085
Tournament Selection

Gradient Search Parameters
Learning Rate = 0.50
Momentum Rate = 0.5
Logistic = 1.0 Number of Iteration = 1000

Net Topology	3:6:6:1	3:6:9:1	3:9:6:1
End Error(GA's)	0.06287	0.0635	0.0560
End Error(B.P.)	0.05270	0.05920	0.04560
Best Ever $^{\text{in gen}}$	369	298	280
Error$^{\text{learn}}_{\text{rms}}$ %	5.26	6.07	4.24
Correlation$^{\text{learn}}$	0.87546	0.8606	0.90130
Deviation$^{\text{learn}}_{\text{stand}}$	123.02	134.48	108.05
Error$^{\text{test}}_{\text{rms}}$ %	4.23	4.590	3.31
Correlation$^{\text{test}}$	0.90631	0.9070	0.93360
Deviation$^{\text{test}}_{\text{stand}}$	107.311	107.57	92.169

Table 4.14: Variation 3: Integration with GA's for test problem 1

Step Size = 0.5			
No.of Mem. Funct	2	3	4
End Error	0.0695	0.0630	0.63280
$Error_{rms}^{learn}$ %	3.78	3.39	3.41
$Correlation^{learn}$	0.9085	0.92550	0.9249
$Deviation_{stand}^{learn}$	105.55	95.66	96.03
$Error_{rms}^{test}$ %	3.80	4.96	2.69
$Correlation^{test}$	0.9177	0.8777	0.8760
$Deviation_{stand}^{test}$	107.803	126.904	127.85

Table 4.15: Variation 4.1 Integration with Fuzzy Logic for test problem 1

Step Size = 0.5

# of Mem. Funct.	5	6	8
End Error	0.06138	0.05928	0.05490
$Error_{rms}^{learn}$ %	3.26	3.139	2.84
$Correlation^{learn}$	0.92956	0.93447	0.9439
$Deviation_{stand}^{learn}$	93.15	89.96	83.42
$Error_{rms}^{test}$ %	3.25	7.930	11.23
$Correlation^{test}$	0.94294	0.7555	0.8437
$Deviation_{stand}^{test}$	86.92	223.90	272.214

Table 4.16: Variation 4.1 Integration with Fuzzy Logic for test problem 1

Step Size = 0.5		
# of Mem. Funct.	2	3
End Error	0.0622	0.0340
$Error_{rms}^{learn}$ %	3.26	1.49
$Correlation^{learn}$	0.9304	0.983
$Deviation_{stand}^{learn}$	92.59	45.72
$Error_{rms}^{test}$ %	2.98	169.7
$Correlation^{test}$	0.9530	----
$Deviation_{rms}^{test}$	79.32	$\sim 10^3$

Table 4.17: Variation 4.2 Integration with Fuzzy Logic for test problem 1

GA's Parameters :

Population Size = 50 Max Generation = 100
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.050
Tournament Selection

# of Mem. Funct.	2	3	4
End Error	0.0701	0.0672	0.0620
Best Ever $^{\text{in gen}}$	29	27	61
Error$^{\text{learn}}_{\text{rms}}$ %	3.67	3.54	3.32
Correlation$^{\text{learn}}$	0.91411	0.91876	0.9279
Deviation$^{\text{learn}}_{\text{stand}}$	102.411	99.753	94.60
Error$^{\text{test}}_{\text{rms}}$ %	3.461	3.687	2.50
Correlation$^{\text{test}}$	0.936650	0.921250	0.93521
Deviation$^{\text{test}}_{\text{stand}}$	96.088	102.958	94.979

Table 4.18: Variation 4.1 Integration with Fuzzy Logic And GA's for test problem 1

GA's Parameters :

Population Size = 50 Max Generation = 100
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.050
Tournament Selection

# of Mem. Funct.	5	6	8
End Error	0.0620	0.055	
Best Ever $^{\text{in gen}}$	39	19	
Error$^{\text{learn}}_{\text{rms}}$ %	3.32	3.11	
Correlation$^{\text{learn}}$	0.92790	0.93452	
Deviation$^{\text{learn}}_{\text{stand}}$	94.160	89.930	
Error$^{\text{test}}_{\text{rms}}$ %	3.47	4.789	
Correlation$^{\text{test}}$	0.93520	0.8838	
Deviation$^{\text{test}}_{\text{stand}}$	94.979	127.128	

Table 4.19: Variation 4.1 Integration with Fuzzy Logic And GA's for test problem 1

GA's Parameters :

Population Size = 50 Max Generation = 100
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.085
Tournament Selection

# of Mem. Funct.	2	3
End Error	0.055730	0.031440
Best Ever $^{\text{in gen}}$	33	16
Error$^{\text{learn}}_{\text{rms}}$ %	2.90	1.58
Correlation$^{\text{learn}}$	0.90172	0.98200
Deviation$^{\text{learn}}_{\text{stand}}$	84.588	47.73
Error$^{\text{test}}_{\text{rms}}$ %	41.42	78.53
Correlation$^{\text{test}}$	0.2050	0.10940
Deviation$^{\text{test}}_{\text{stand}}$	1196.28	2291.73

Table 4.20: Variation 4.2 Integration with Fuzzy Logic And GA's for test problem 1

Results - Pachuca Tank Problem

TABLE: FOR SCALING PACHUCA TANK

Parameters	Minimum	Maximum
U_g (m/s)	0.1	1.5
D_g/D_t	0.5	1.0
H_v/D_t	0.5	5.0

Table 4.21 Table used for scalling for test problem 2

Learning Rate : 0.50 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:1:1	4:2:1	4:3:1
End Error	0.063181	0.06041	0.06284
$Error_{rms}^{learn}$ %	16.43	14.30	14.32
$Correlation^{learn}$	0.91651	0.9246	0.9249
$Deviation_{stand}^{learn}$	0.52566	0.4965	0.49765
$Error_{rms}^{test}$ %	23.59	18.43	18.42
$Correlation^{test}$	0.96437	0.95364	0.95387
$Deviation_{stand}^{test}$	0.4000	0.394766	0.391022

Table 4.22: RESULT BP Variation 1.1 for test problem 2

Learning Rate : 0.50 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:4:1	4:5:1	4:1:1:1
End Error	0.057635	0.057651	0.06581
$Error_{rms}^{learn}$ %	14.021	14.040	17.790
$Correlation^{learn}$	0.92736	0.92749	0.91338
$Deviation_{stand}^{learn}$	0.4909	0.4907	0.91338
$Error_{rms}^{test}$ %	18.851	18.727	21.147
$Correlation^{test}$	0.93345	0.95240	0.94801
$Deviation_{stand}^{test}$	0.41675	0.402981	0.4201

Table 4.23: RESULT BP Variation 1.1 for test problem 2

Learning Rate : 0.50 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:1:1	4:2:1	4:3:1
End Error	0.0792	0.0828	0.0921
$Error_{rms}^{learn}$ %	41.35	34.66	73.24
$Correlation^{learn}$	0.9063	0.9111	0.8860
$Deviation_{stand}^{learn}$	0.729	0.677	1.007
$Error_{rms}^{test}$	59.73	42.82	81.71
$Correlation^{test}$	0.9574	0.9580	0.9444
$Deviation_{stand}^{test}$	0.6699	0.5560	0.8757

Table 4.24: BP Variation 1.2 for test problem 2

Learning Rate : 0.50 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:4:1	4:5:1	4:1:1:1
End Error	0.9000	0.7179	0.1857
$Error_{rms}^{learn}$ %	72.0077	72.057	129.53
$Correlation^{learn}$	0.90815	0.878	0.9113
$Deviation_{stand}^{learn}$	0.9393	1.0199	1.953
$Error_{rms}^{test}$ %	86.54	64.34	198.74
$Correlation^{test}$	0.958	0.9090	0.9617
$Deviation_{stand}^{test}$	0.8841	0.8691	2.264

Table 4.25: BP Variation 1.2 for test problem 2

Learning Rate : 0.05 Momentum : 0.05

Logistic = 1.00
Number of Iteration = 1000

Net Topology	4:4:1	4:5:1	4:1:1:1
End Error	0.0596	0.0588	0.05694
$Error_{rms}^{learn}$ %	15.726	15.27	16.09
$Correlation^{learn}$	0.91568	0.9183	0.9162
$Deviation_{stand}^{learn}$	0.5413	0.5275	0.540
$Error_{rms}^{test}$ %	14.48	15.79	18.63
$Correlation^{test}$	0.9724	0.968	0.973
$Deviation_{stand}^{test}$	0.3567	0.3605	0.378

Table 4.26: BP Variation 1.3 for test problem 2

Learning Rate : 0.05 Momentum : 0.05

Logistic = 1.00
Number of Iteration = 1000

Net Topology	4:1:1	4:2:1	4:3:1
End Error	0.0592	0.0671	0.0595
$Error_{rms}^{learn}$ %	15.68	16.55	15.373
$Correlation^{learn}$	0.9164	0.9124	0.9159
$Deviation_{stand}^{learn}$	0.5395	0.5516	0.5409
$Error_{rms}^{test}$ %	16.26	21.53	16.91
$Correlation^{test}$	0.973	0.9549	0.9369
$Deviation_{stand}^{test}$	0.3668	0.4002	0.3775

Table 4.27: BP Variation 1.3 for test problem 2

Learning Rate : 0.50 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:1:1	4:2:1	4:3:1
End Error	0.06236	0.5933	0.065
$Error_{rms}^{learn}$ %	20.32	15.94	15.44
$Correlation^{learn}$	0.91618	0.91598	0.9169
$Deviation_{stand}^{learn}$	0.57235	0.5093	0.5200
$Error_{rms}^{test}$ %	27.31	20.34	18.985
$Correlation^{test}$	0.9694	0.97047	0.97114
$Deviation_{stand}^{test}$	0.4514	0.3445	0.3529

Table 4.28: BP Variation 1.4 for test problem 2

Learning Rate : 0.50 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:4:1	4:5:1	4:1:1:1
End Error	0.0592185	0.0642	0.06496
$Error_{rms}^{learn}$ %	21.61	15.20	17.01
$Correlation^{learn}$	0.9184	0.9184	0.913
$Deviation_{stand}^{learn}$	0.5813	0.5087	0.525
$Error_{rms}^{test}$ %	23.099	16.77	25.56
$Correlation^{test}$	0.9686	0.97039	0.9644
$Deviation_{stand}^{test}$	0.4447	0.3251	0.3911

Table 4.29: BP Variation 1.4 for test problem 2

Learning Rate : 0.30 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:1:1	4:2:1	4:3:1
End Error	0.05959	0.05969	0.05821
$Error_{rms}^{learn}$ %	16.97	16.97	15.84
$Correlation^{learn}$	0.9124	0.9123	0.9166
$Deviation_{stand}^{learn}$	0.54629	0.5458	0.5354
$Error_{rms}^{test}$ %	21.53	21.59	15.74
$Correlation^{test}$	0.9687	0.9689	0.9672
$Deviation_{stand}^{test}$	0.3957	0.3949	0.3695

Table 4.30: BP Variation 1.5 for test problem 2

Learning Rate : 0.30 Momentum : 0.50

Logistic = 0.70
Number of Iteration = 1000

Net Topology	4:4:1	4:5:1	4:1:1:1
End Error	0.0586	0.0578	0.0639
$Error_{rms}^{learn}$	16.07	15.57	18.80
$Correlation^{learn}$	0.9155	0.9174	0.9064
$Deviation_{stand}^{learn}$	0.53748	0.5333	0.5427
$Error_{rms}^{test}$	16.783	14.95	30.12
$Correlation^{test}$	0.96824	0.9687	0.9638
$Deviation_{stand}^{test}$	0.3719	0.35955	0.379299

Table 4.31: BP Variation 1.5 for test problem 2

Number of Center	2	4	6
$Error^{learn}$	0.07606	0.07580	0.07420
$Error^{learn}_{rms}$ %	29.51	29.630	25.701
$Correlation^{learn}$	0.84780	0.85170	0.871770
$Deviation^{learn}_{stand}$	0.70960	0.70150	0.6870
$Error^{test}_{rms}$ %	54.50	63.88	27.135
$Correlation^{test}$	0.9148	0.8969	0.9684
$Deviation^{test}_{stand}$	0.65760	0.7520	0.3973

Table 4.32: RBF network Variation 2 for test problem 2

Number of Center	8	10	12
$Error^{learn}$	0.07090	0.06034	0.05670
$Error^{learn}_{rms}$ %	28.74	25.570	17.02
$Correlation^{learn}$	0.87177	0.90900	0.9201
$Deviation^{learn}_{stand}$	0.65777	0.56230	0.52930
$Error^{test}_{rms}$ %	59.15	80.106	55.64
$Correlation^{test}$	0.8923	0.85802	0.9038
$Deviation^{test}_{stand}$	0.70040	0.956949	0.6902

Table 4.33: RBF network Variation 2 for test problem 2

GA's Parameters :

Population Size = 1000 Max Generation = 500
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.085
Tournament Selection

Gradient Search Parameters

Learning Rate = 0.50
Momentum Rate = 0.50
Logistic = 1.0 Number of Iteration = 1000

Net Topology	4:1:1	4:2:1	4:3:1
End Error(G.A)	0.07319	0.06966	0.07335
End Error(BP)	0.064242	0.06342	0.0649
Best Ever $^{\text{in gen}}$	56	32	300
Error $^{\text{learn}}_{\text{rms}}$ %	17.77	16.88	17.92
Correlation $^{\text{learn}}$	0.9123	0.9158	0.9108
Deviation $^{\text{learn}}_{\text{stand}}$	0.5177	0.5069	0.5264
Error $^{\text{test}}_{\text{rms}}$ %	27.93	24.63	28.79
Correlation $^{\text{test}}$	0.9505	0.9580	0.96343
Deviation $^{\text{test}}_{\text{stand}}$	0.4415	0.39612	0.4172

Table 4.34 : Variation 3: Integration with GA's for test problem 2

GA's Parameters :

Population Size = 1000 Max Generation = 500
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.085
Tournament Selection

Gradient Search Parameters
Learning Rate = 0.50
Momentum Rate = 0.5
Logistic = 1.0 Number of Iteration = 1000

Net Topology	4:4:1	4:5:1	4:1:1:1
End (GA's)	0.07476	0.06874	0.07609
End Error (B.P.)	0.07099	0.06496	0.07042
Best Ever $^{\text{in gen}}$	261	59	81
Error $^{\text{learn}}_{\text{rms}}$ %	21.64	19.17	20.57
Correlation $^{\text{learn}}$	0.8936	0.914041	0.9010
Deviation $^{\text{learn}}_{\text{stand}}$	0.5861	0.51900	0.5650
Error $^{\text{test}}_{\text{rms}}$ %	35.98	32.36	35.48
Correlation $^{\text{test}}$	0.9505	0.95051	0.9556
Deviation $^{\text{test}}_{\text{stand}}$	0.4830	0.4689	0.4881

Table 4.35 : Variation of integration with GA's for test problem 2

Step Size = 0.5

# of Mem. Funct.	2	3	4
End Error	0.04671	0.02652	0.041250
$Error_{rms}^{learn}$ %	11.31	7.73	11.23
$Correlation^{learn}$	0.9727	0.99130	0.97880
$Deviation_{stand}^{learn}$	0.316110	0.17979	0.27910
$Error_{rms}^{test}$ %	54.55	$\sim 10^{6}$	$\sim 10^{3}$
$Correlation^{test}$	0.60306	0.2740	0.1550
$Deviation_{stand}^{test}$	1.199	$\sim 10^{3}$	52

Table 4.36: Variation 4.1 Integration with Fuzzy Logic for test problem 2

GA's Parameters :

Population Size = 50 Max Generation = 100
Probablity of Crossover = 0.90 (cross over sight = 4)
Probablity of mutation = 0.050
Tournament Selection

# of Mem. Funct.	2	3	4
End Error	0.03627	0.030990	0.022119
Best Ever $^{\text{in gen}}$	53	68	49
Error$^{\text{learn}}_{\text{rms}}$ %	9.930	9.34	6.902
Correlation$^{\text{learn}}$	0.98630	0.98812	0.99319
Deviation$^{\text{learn}}_{\text{stand}}$	0.24560	0.21001	0.15001
Error$^{\text{test}}_{\text{rms}}$ %	21.33	41.52	636.7
Correlation$^{\text{test}}$	0.955721	0.8889	0.5250
Deviation$^{\text{test}}_{\text{stand}}$	0.4417530	0.55787	16.42

Table 4.37: Variation 4.1 Integration with Fuzzy Logic And GA's for test problem 2

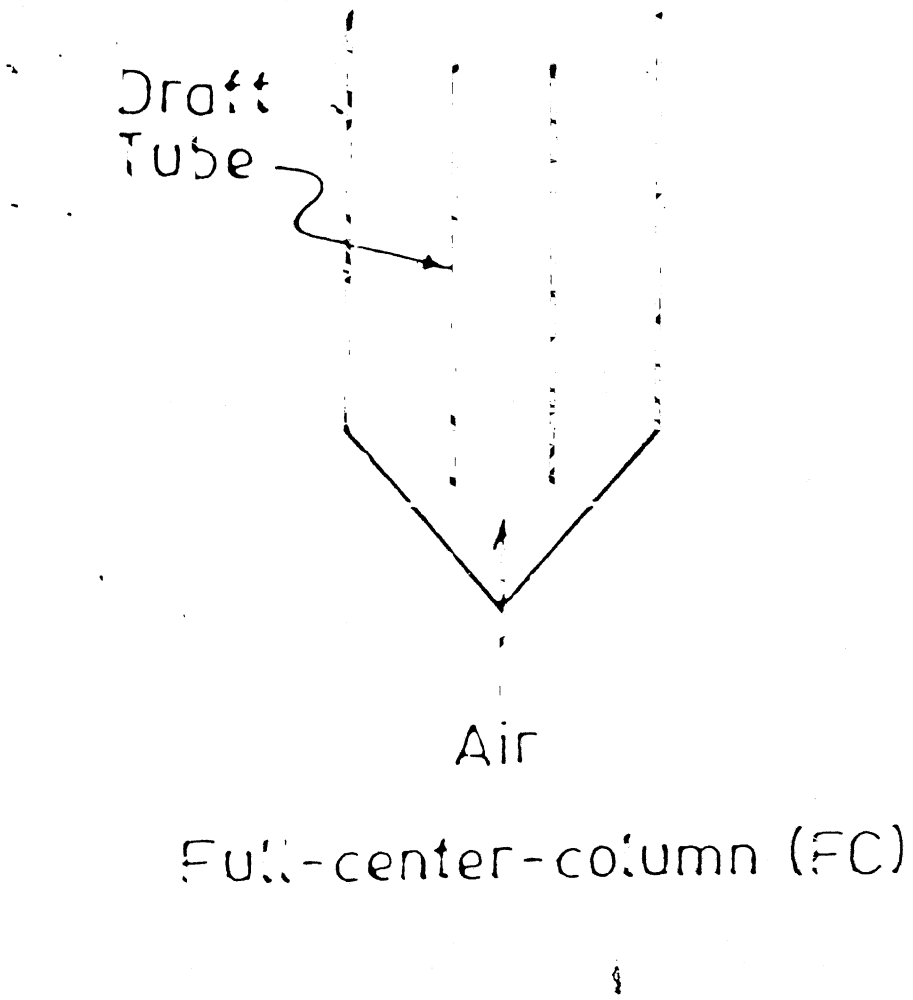


Figure 4.1: Pachuca Tank (FCC)

Main Results - Tables And Figures

TABLES FOR BEST RESULT FOR TEST PROBLEM NO. 1

Variation of ANN	RMS Error in Prediction %	Correlation Coeficient	Standard Deviation	Comments
BP and It's Variations	3.314	0.93724	92.20	Randomize Pattern
Integration with GA's	3.3215	0.93868	93.50	GA'S is used to optimize weights
Radial Basis Function	5.30840	0.865662	135.32	Speed of Learning is Very Fast
Integration with Fuzzy Logic	2.69	0.8760	127.85	Second Best Result
Integration with Fuzzy Logic and GA's	2.50	0.921250	94.973	Best Result

Table 4.38. RESULTS for test problem 1

TABLES FOR BEST RESULT FOR TEST PROBLEM NO. 2

Variation of ANN	RMS Error in Prediction %	Correlation Coeficient	Standard Deviation	Comments
BP and It's Variations	14.48	0.97247	0.3567	Changed Activation Function (Best Result)
Integration with GA's	24.63	0.9580	0.39612	GA'S is used to optimize weights
Radial Basis Function	27.135	0.9684	0.3973	Speed of Learning is Very Fast
Integration with Fuzzy Logic	54.55	0.60306	1.99	Problem Due to Overlearning
Integration with Fuzzy Logic and GA's	21.33	0.955721	0.4417530	Problem Due to Overlearning

Table 4.39 RESULTS for test problem 2

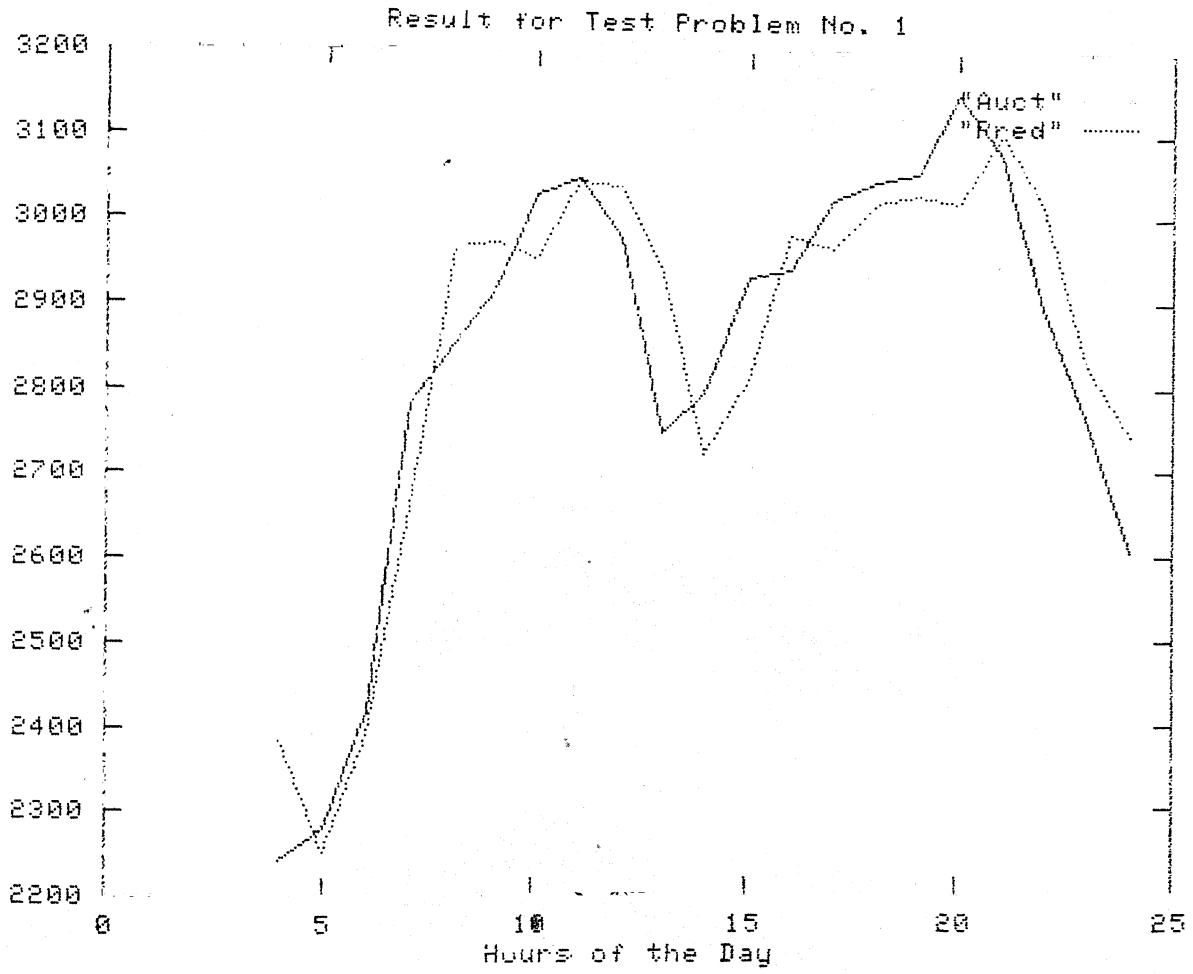


Figure 4.2: RESULTS for test problem

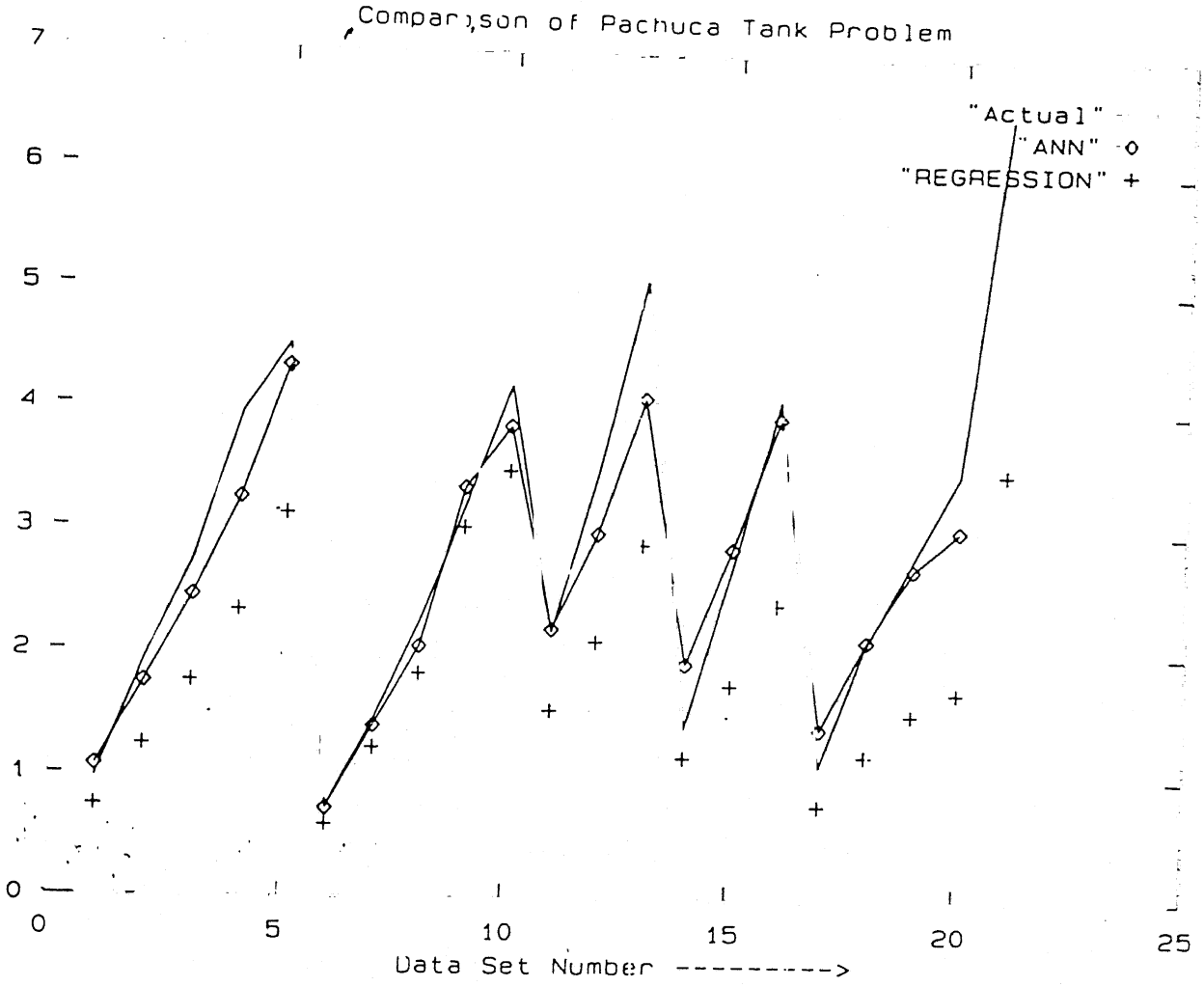


Figure 4.3: RESULTS for test problem 2

Chapter 5

Conclusions and Recommendation for future work

5.1 Conclusions

It is found that the sucess of BP and it's variation is problem dependent. Among back-propagation and it's four variations, random presentation of patterns in each epoch work better in one of the variation (test problem No. 1). The training time required for all the variations are approximately the same. Use of logarithmic error function in BP does not prove to be a good alternative to RMS error function. Logarithmoid nonlinear filter works better than BP with preprocessed/scaled data and it produces the best result in test problem No. 2. Proportionate error learning does not improve the learning speed.

Radial basis function is a good alternative to BP. The learning time of RBF is several orders less than BP.

Integration of genetic algorithms with BP does not work good for both the test problems. The learning time is much - much higher than that of BP while prediction results are comparable to that of BP. The effect of different parameters of Genetic algorithm for example crossover probability, mutation probability, string length, different types of selection

schemes and crossover, is not studied in this present work. The failure of GA's assisted BP may be attributed to the fact that the search space become limited by applying GA's.

Integration of fuzzy logic with neural network that is Adaptive Network Fuzzy Inference System (ANFIS) gives good results for one of the problems. In the case of second test problem it suffers the problem of overlearning (less datas where avaliable for this test problem). Time required with this algorithm is much less than that of backpropagation and its variations. In ANFIS the adaptive parameters selected by the user is much less than that in the case of BP. However, the number of adjustable parameters (weights) increases rapidly with increasing number of rules for each input. This results in overlearning of bigger networks of ANFIS.

Integration of GA's with ANFIS to train the network gives better results than ANFIS for both test problems. The success of genetic algorithm in case of ANFIS may be attributed to the fact that the premise parameters range (for which GA's is used) is properly selected. Data scaling in ANFIS and GA's assisted ANFIS is not needed. This may be useful in solving many problems having difficulty of scaling ranges and type of scaling used. The prediction results of GA's assisted ANFIS is best among all the algorithms tried in this work for test problem No. 1. In the case of the second problem, as mentioned above less training samples produce problem of over learning.

5.2 Recommendation for Future Work

- Other variations of backpropagation needs to be tried for speeding up the algorithm. Other nonlinear filter may be used to speed up the learning and performance.
- In RBF, the other clustering algorithms as fuzzy clustering should be tried.
- Effect of GA parameters on the learning of the network should be studied.
- Effect of different And/Or operation in ANFIS needs to be studied.

Bibliography

[1] Rich E. and Knight K. (1991), *Artifical Intelligence*, second edition, Tata McGraw - Hill.

[2] Goldberg D.E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley,Reading,Mass.

[3] Davis L. (1991), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

[4] Lippmann. R. P. (1987), *An Introduction to Computining with Neural nets*, IEEE ASSP Magazine., V-4,pp. 4-22.

[5] Narendra K. and Thathachar (1983), *Learning Automata — an intoduction*, Prentice – Hall International.

[6] Zadeh, L. A. (1988), *Fuzzy Logic* , Computers, Vol. 21,pp. 83-92.

[7] Gelfand B. S. and Delp E. J. (1991), *On Tree Structured Classifiers*, Artifical Neural Networks,Elsevier Science Publishers B.V.

[8] Codd E. F. (1963), *Cellular Automata*, Academic Press.

[9] Wasserman P. D. (1989), *Neural Computing Theory and Practice*, Van Nostrand Reinhold, New York.

[10] . Fu LiMin (1994), *Neural Networks in Computer Intelligence*, McGRAW - HILL. .

[11] Pao Y. H. (1989), *Adaptive Pattern Recognition and Neural Networks* Addison-Wesley.

[12] Sugeno M., (1985), *An introductory Survey of Fuzzy Control*, Information Sciences, Vol. 36, pp 59 - 83,1985.

[13] Brubaker D., (1994), *Dismantling a rule-based system shows its inner working*, EDN Nov 10, 1994, pp 189 – 193.

[14] Kir G. J. and Folger T. A. (1994), *Fuzzy Sets Uncertainty, and Information*, Prentice – Hall of India Limited.

[15] Jang Roger S.J.,(1993), *ANFIS: Adaptive - Network - Based Fuzzy Inference System* , IEEE Trans on SMC, Vol. 23,No. 3, pp. 655-685.

[16] Filho J.L.R. and Treleaven C. (1994), *Genetic – Algorithm Programming Environments*, Computers,June 94.

[17] Buckley J.J. and Hayasi Y. Fuzzy genetic algorithm and it's applications, Fuzzy Sets and Systems, 1994, pp. 129-136.

[18] Hush D. and Horne B. G., (1993), *Progress in Supervised neural Networks (What New Since Lippmann)*, IEEE Signal Processing Magazine.

[19] Simpson P. K., (1990), *Foundation of Neural Networks*, Neural Networks, Vol 6, pp. 595-603, 1990.

[20] Ranaweera D.K. and Karady G.(1993), *Power System Static Analysis using Radial Basis Function Neural Network*, ESP 93,pp. 272-274.

[21] Abhay Bulsari et.al , (1993), *Investigation of The Systemmetric Logarithmoid As an Activation Function For Neurans In Feed Forward Neural Networks*,, Artifical Neural Networks

[22] M.T. Musavi et al., (1992), *On Traning of radial Basis Function Classifier*, Neural Networks, Vol 5, pp. 595-603.

[23] Leonard J.A. and Kramer M. A., (1991), *Radial Basis Function Networks for Classifying Process Faults* , IEEE Control Systems.

[24] Zadeh L.A., (1973), *Outline of a new approach to the analysis of complex systems and decision processiess*, IEEE Trans. Sys. Man and Cybern., vol 212,1973, pp 28 - 44.

[25] Ooyen A. Van, (1992),*Improving the Convergence of the Back-Propagation Algorithm*, Neural Networks,Vol 5,pp. 465-471.

[26] Burton R. M. (1992), *Event dependent Control of Noise Enhances Learning in Neural Networks*, Neural Networks, Vol 5, pp. 627-637.

[27] Janson D.J. and Frenzel F. (1993),*Training Product Unit Neural Networks with Genetic Algorithms* ,IEEE EXPERT, Octber 1993.

[28] Tim H. J. J. Van Der Hagen (1994), *The Scaling Parameter of the Sigmodial Function in Artifical Neural Networks*, Nuclear technology, Vol 106, pp. 135-138, April 1994.

[29] Christian Cachin (1994), *Pedagogical Pattern Selection Strategies* Neural networks, vol 7 No 1, pp. 175-181, 1994.

[30] Reed R., (1993), Pruning Algorithms - A Survey, IEEE Transactions On Neural Networks Vol 4, No. 5, Sept.

[31] Leonard A. J. and Kramer M. A.,(1991), *Radial BAssis Function Networks for Classifying Process Faults*, IEEE Control System, pp - 31-38, April 1991.

[32] Musavi M. T. et al.(1992), *On Traning of Radial Basis Function Classifier*, Neural networks, vol 5 , pp. 595-603, 1992.

[33] Ranaweera D. K. and Karady G.(1989), *Power System Static Security Analysis using Radial Bassis Function Neural Network*, ESAP-89, pp 272-274.

[34] Eshelman L. E. et. al., (1989), *Biases in the Crossover Landscape*, ICGA-89, pp 10-19.

[35] Rumelhart, D. E. et.al. (1986), *Parallel Distributed Processing* Vol 1, MIT Press, Cambridge, MA.

[36] Caudill M. A.(1991), *Neural Networks Training Tips and techniques*, AI Expert, Jan 1991, pp. 56-61.

[37] G. G. Roy, (1995), *PHD Thesis* , IIT Kanpur.

[38] Srinivasan D. et.al, (1994), *A Neural Network Short–Term Load Forecaster*, Electric Power System Research, vol 28, pp 227 - 234.

Appendix A

Radial Basis Function Networks

The Radial Basis function (Moody and Darken) network is a two layered network where the output unit form a linear combination of the basis (kernel) functions computed by hidden layer. The most common basis function is Gaussian function (used in this simulator) in which the activation level O_j is calculated as $\exp(-\frac{x-\bar{x}^2}{\sigma*\sigma})$ Where $\bar{x} = center\ of\ the\ basis\ function$
Where $\sigma = width\ of\ the\ basis\ function$
The output can be calculated by $\sum w_j O_j$
where
O_j is the output of the hidden layer
W_j is the associated weight

Thus the output level is a linear combination of nonlinear basis function.

The heart of this algorithm is to calculate $\bar{x}$, the clustering center of input attributes. This simulator uses K-mean algorithm to calculate center of cluster. The K-mean algorithm can be represented as :

The center of each cluster is initialized to a different randomly selected training pattern. Then, each training pattern is assigned to the cluster closed to it. This can be done by calculating the Euclidean distance between the training pattern and all the cluster centers.

When all the training pattern is assigned, the average position of the cluster is calculated and the cluster center is moved to that point. This process is repeated until algorithms converges. Each cluster is associated with one of the RBF's in the network and the cluster centers become the unit center $\bar{x}$ of RBF's. If X_i is one center and X_j is the nearest center from it the σ is calculate as $\sigma = \sqrt{\sum_{j=1}^{p} (X_j - X_i)^2}$. In this case the value of p is equal to one.

As output from gaussian function and real output is known so weight can be calculated by least square method [20] - [23] These center and weights are used in prediction.

Appendix B

Integration of GA's with ANN

Different Techniques are used to integrate GA's with ANN. In the simulator ANNS GA's is used as an optimization device to minimization the error of BP. As stated earlier in chapter 1 the objective function is given as

$$Fitness = \frac{1}{1+E} \tag{B.1}$$

As output of this function depends upon weights of ANN. For best set of weights the value obtained from this function called fitness should be *1* theocratically.

B.1 Working principle

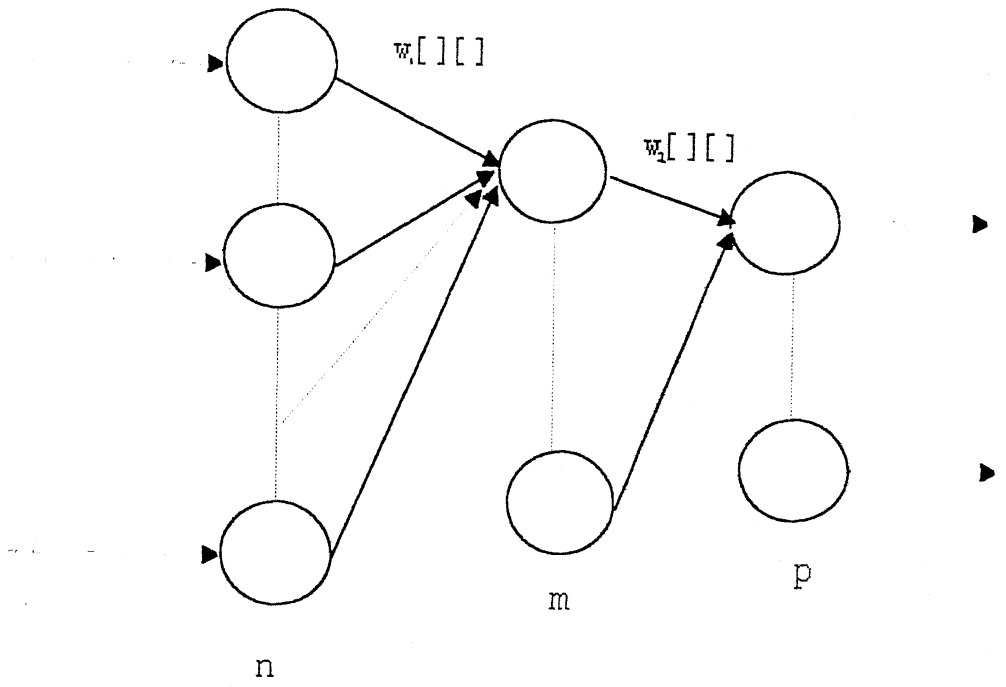
GA's works with population of chromosome where each chromosome represents one complete collection of sets of weights. Length of each chromosome is calculated by the precision required by each weights.

For Example if l represents the number of bits required to calculated for each weight then it can be given by

$$l_l = log_2/[(w_{max} - w_{min}/prec + 1] \tag{B.2}$$

and hence total number of bits in one chromosome = l_w*$total\ number\ of\ weights\ of\ ANN$.

Another important feature is *How we arrange weights in the chromosome.* As for example let us consider the ANN given below



This ANN consist of ten weights and hence to l_w can be arranged in $P^1 0_{10}$ ways. The method adopted in this simulator is the weights directing to one node is kept together. (as it is advisable to keep the all the parameter which influences the outcome). So weights are kept together as shown below

W_{111} W_{121} W_{131}--- W_{1n1} W_{112} W_{122} W_{132}----- W_{1n2}----- W_{1nm} W_{211}---- W_{2mp}

In decoding first the each l_w is decoded into digital value let us say d. Now this digital value is rescaled to obtained the value of weights as

$$W = W_{min} + \frac{W_{max} - W_{min}}{2^{lw} - 1} * d \tag{B.3}$$

B.2 Working Of GA's and Backpropagation

1. Length of chromosome is calculated by getting the information precision (given by user) and number of weights decided by topology of ANN.
2. Generate the random chromosomes equal to population size given by user.
3. Decode the weights and perform forward processing.
4. Calculate the fitness value
5. Perform Different GA's operation
   - Selection
   - Crossover
   - Mutation
6. If best solution is achieved then save it.
7. Repeat the process 3 to 6 until the convergence occurs.

Details Of Step 5

- *Selection Operator*

  There are two different selection are used in th simulator, Roulet Wheel, and Tournament Selection [2].

- Crossover Operator

  There are two type of crossover used as single point and multiple point crossover [16].

NETP-1995-M-PRA-ART